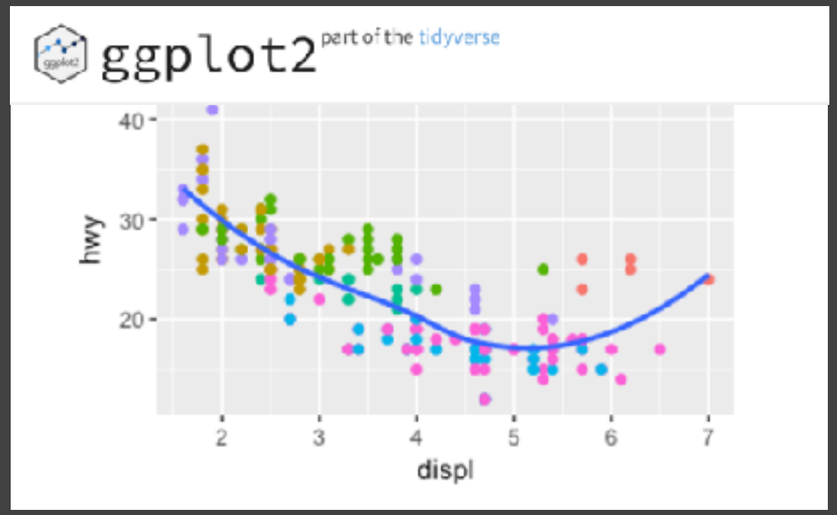
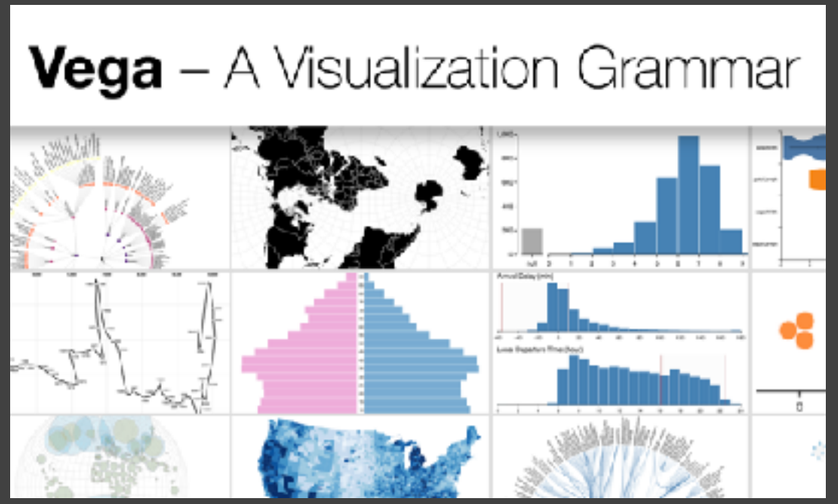
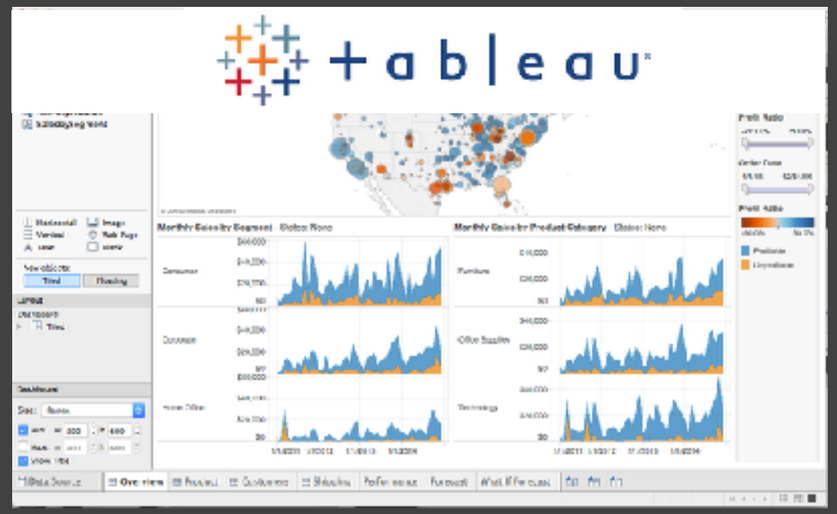
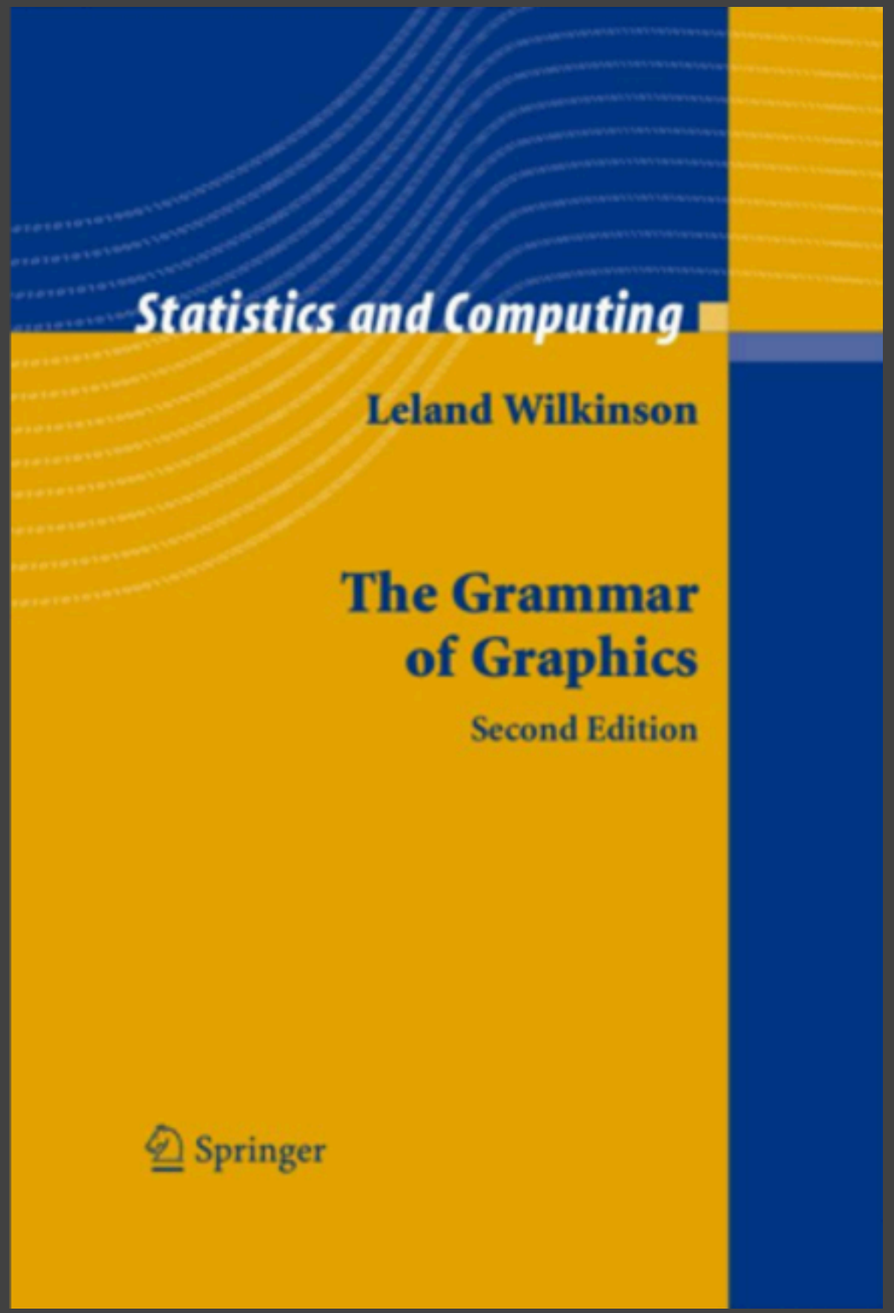
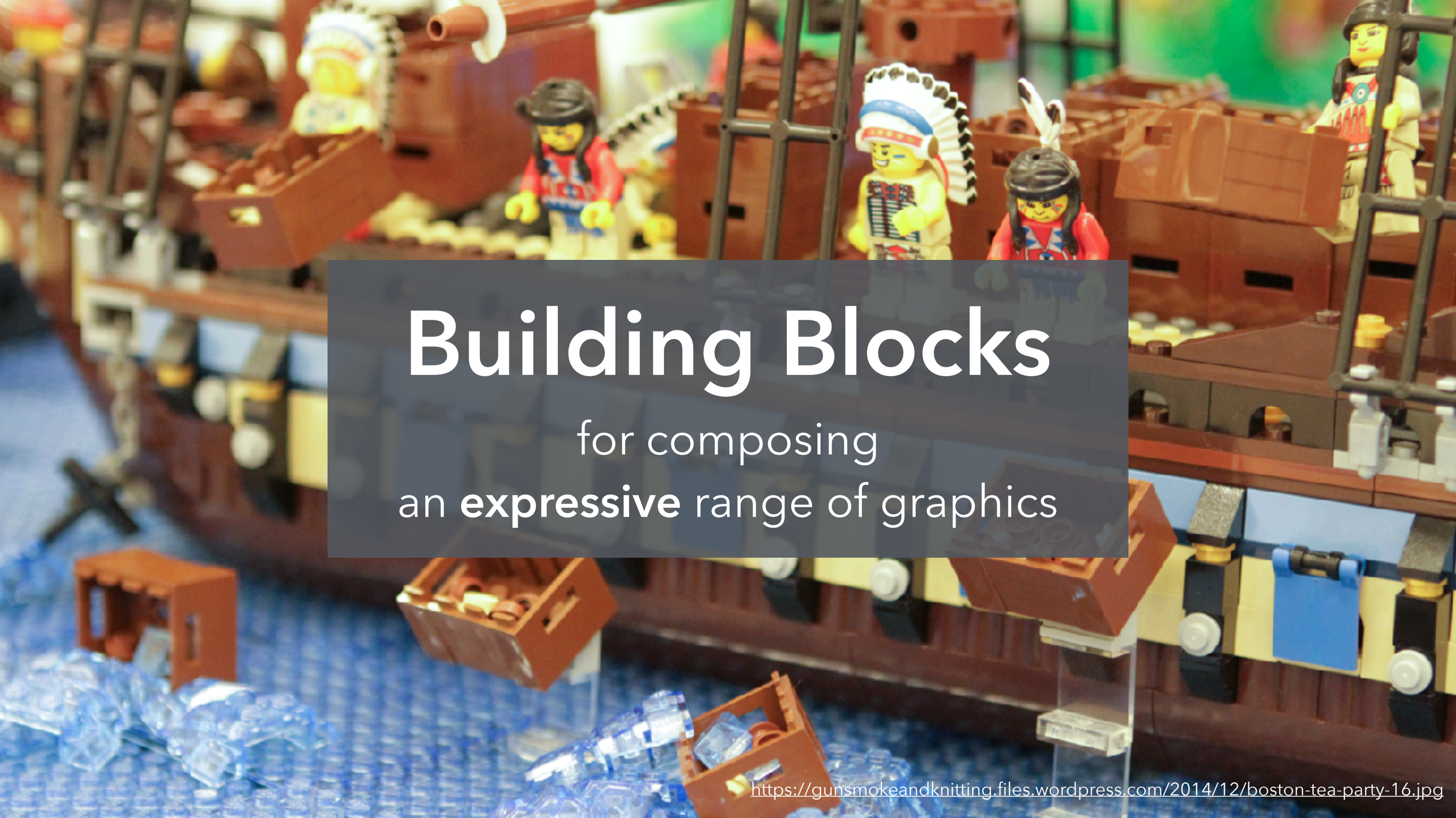

Data-Driven Documents


ggplot2
part of the tidyverse
40
30
20
hwy
2
3
4
5
6
7
displ


Vega – A Visualization Grammar


+ a b | e a u


Statistics and Computing
Leland Wilkinson
The Grammar of Graphics
Second Edition
Springer



Building Blocks

for composing
an **expressive** range of graphics

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

Grammar of Graphics

Data Input data source to visualize.

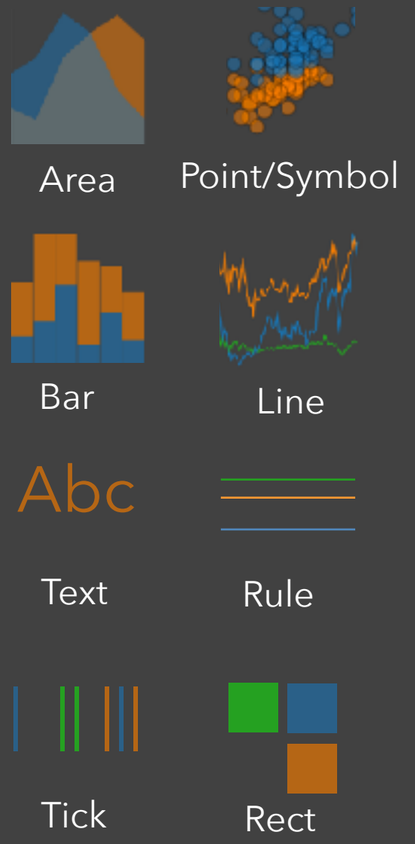
Transform Filter, aggregation, binning, etc.

Mark Data-representative graphics.

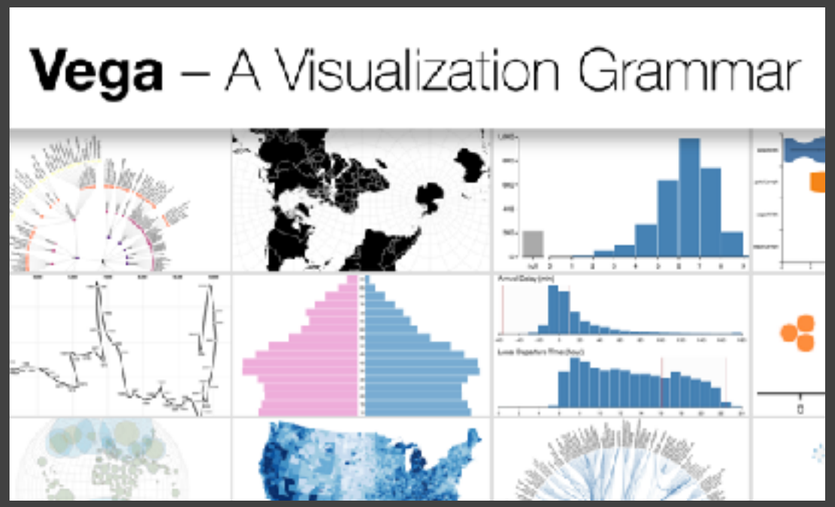
Encoding Mapping between data and mark properties.

Scale Functions that map data values to visual values.

Guides Axes & legends that visualize scales.



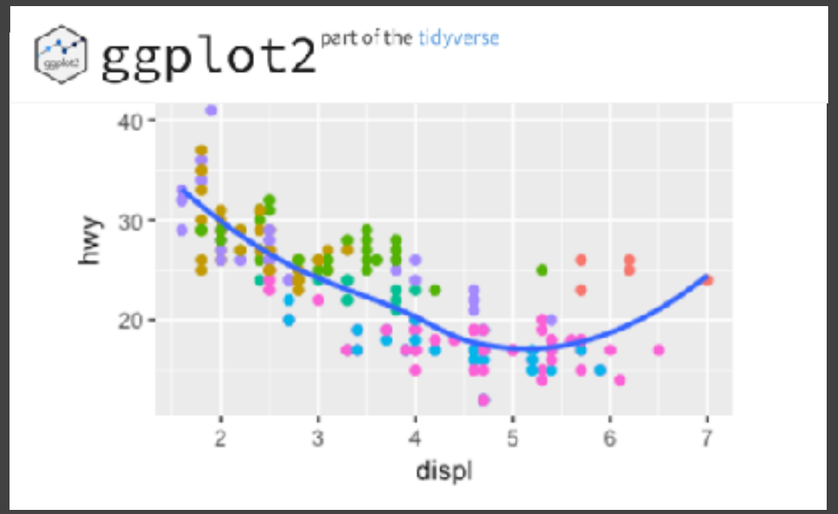
Grammar of Graphics *for Customized Designs*

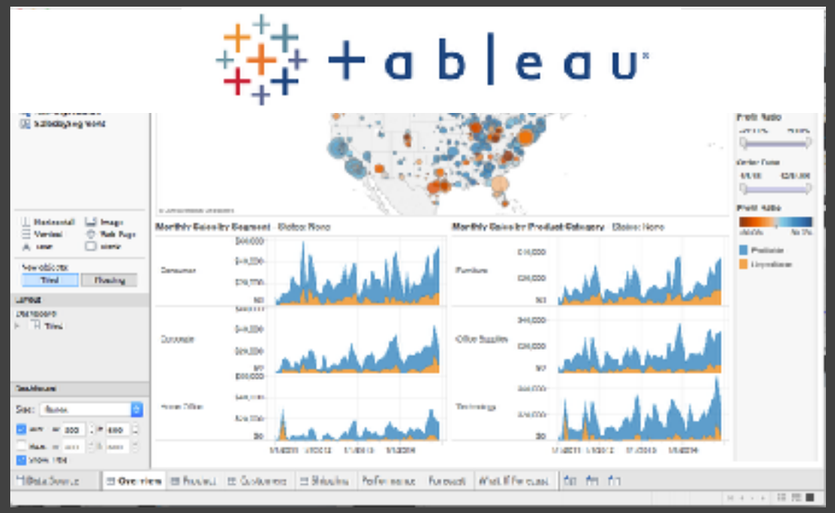


Offer **fine-grained control** for composing interactive graphics.

But require **verbose** specifications and technical expertise.

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.

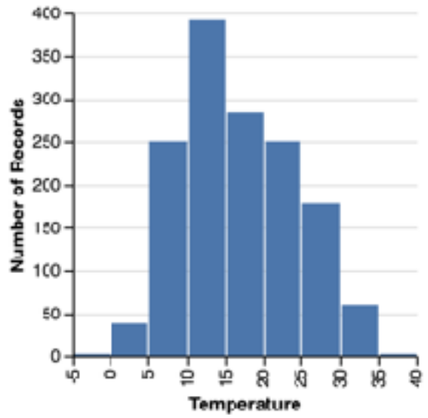
But **limited** support for **interactions.**

Vega-Lite's Mission

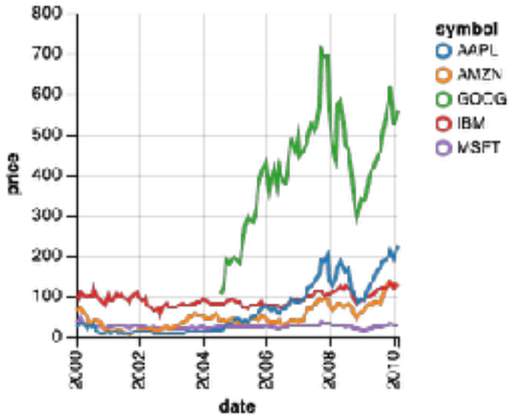
Facilitate exploratory data analysis
with an ***expressive*** yet ***concise*** language
to specify ***interactive multi-view graphics***

Vega-Lite: a Grammar of Graphics

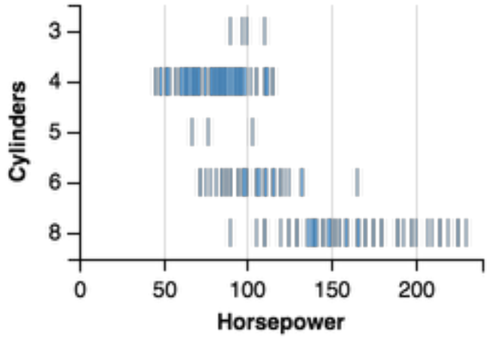
Histogram



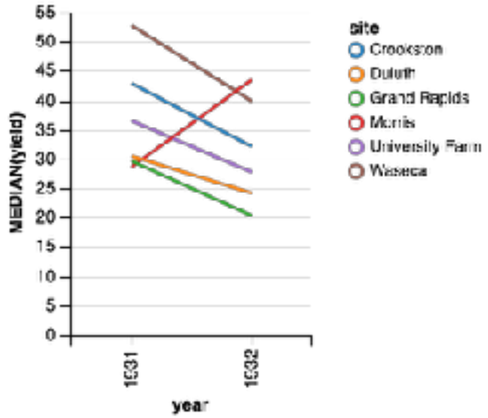
Multi-series Line Chart



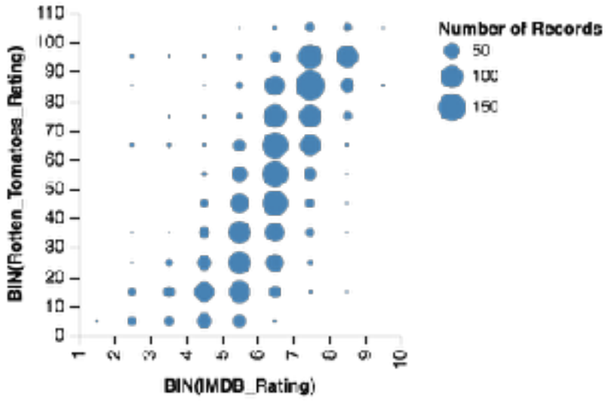
Stripplot



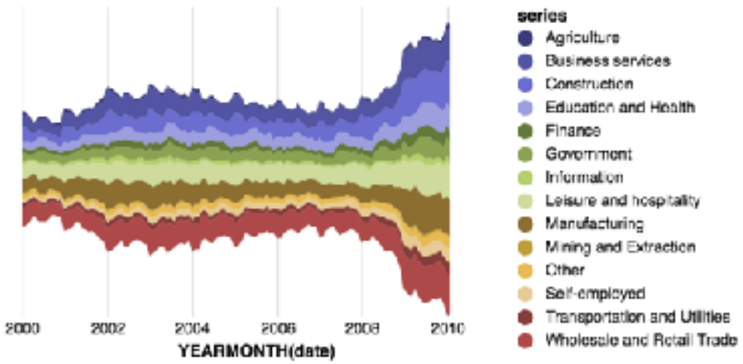
Slope Graph



Binned Scatterplot

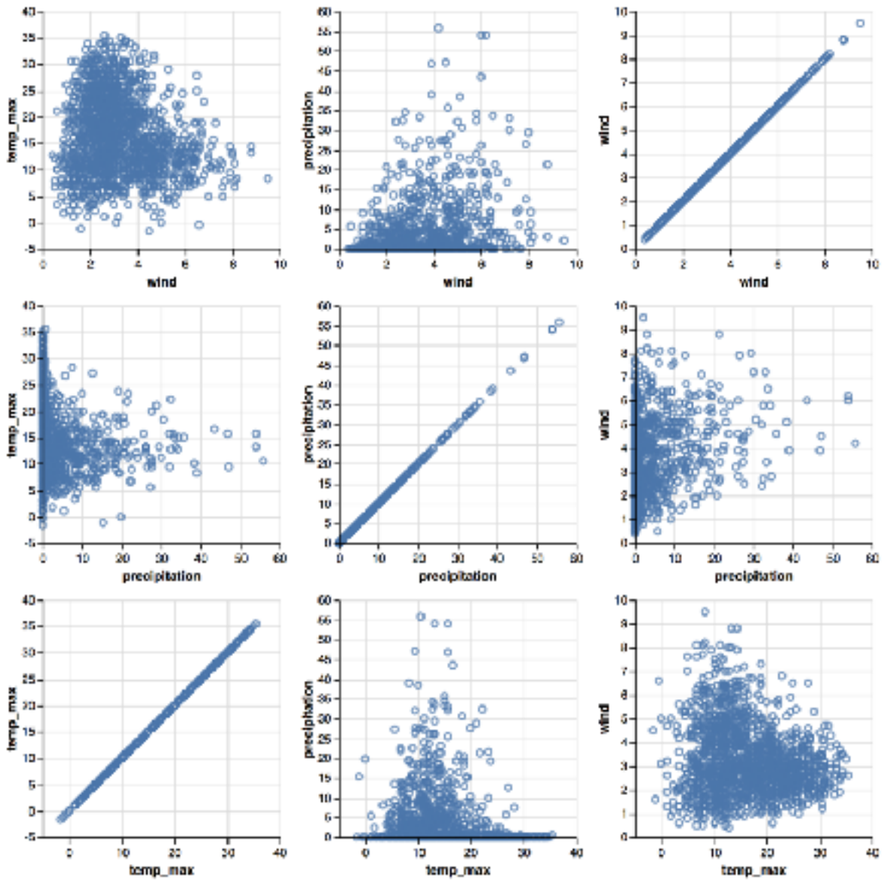


Area Chart



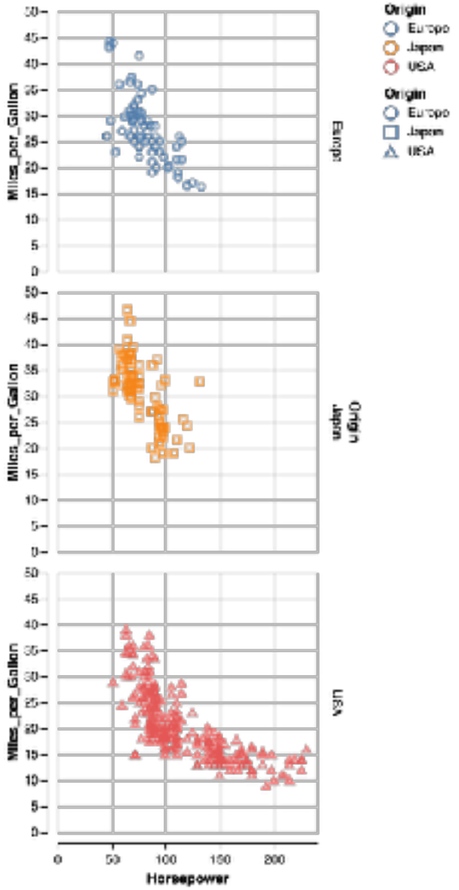
Vega-Lite: a Grammar of Multi-View Graphics

Scatterplot Matrix



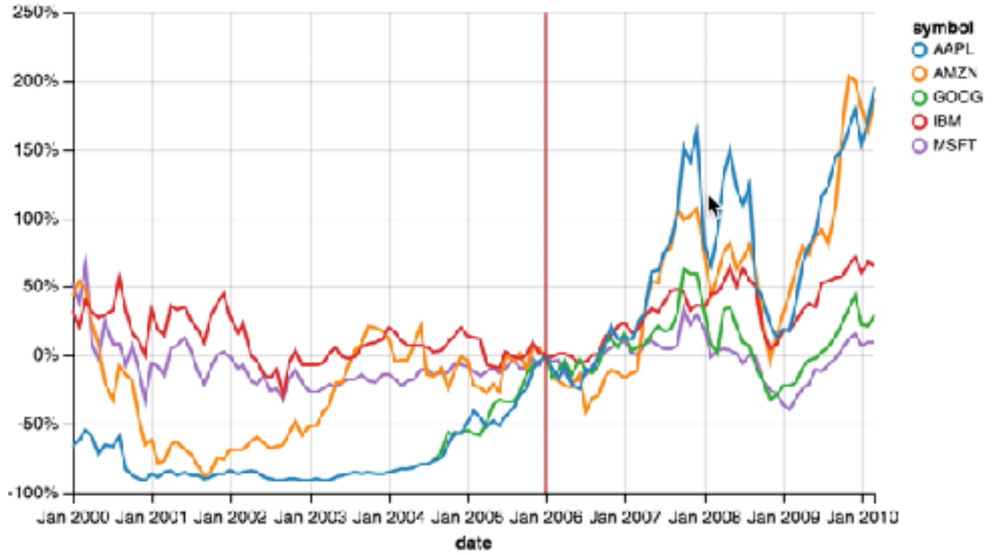
Concatenated & Layered View

Faceted View

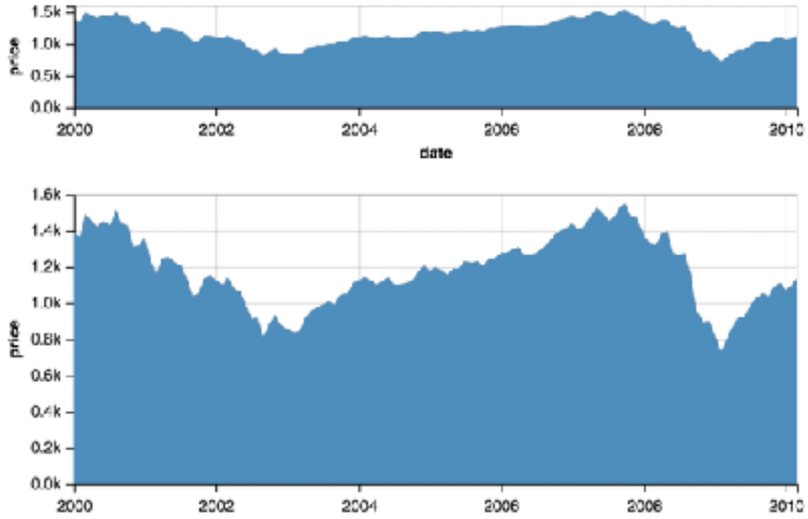


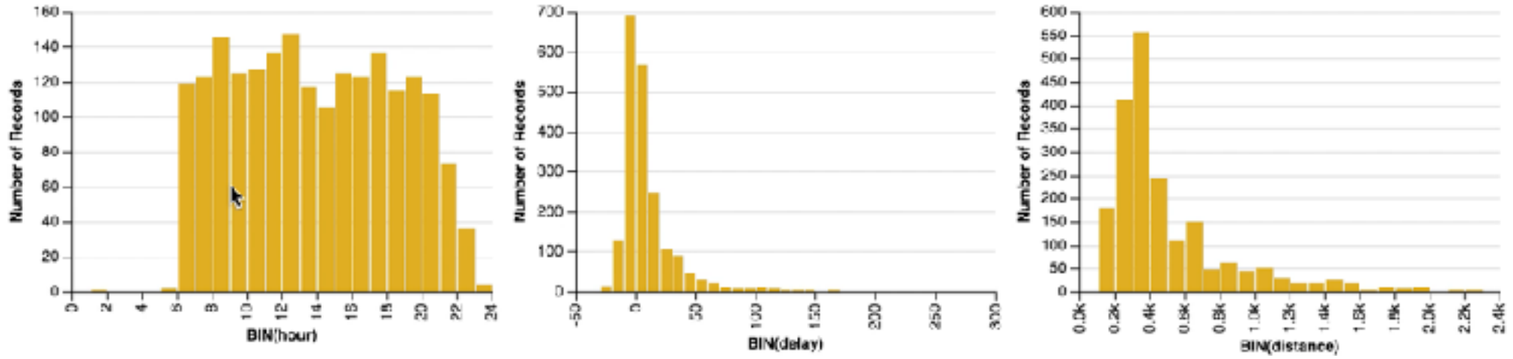
Vega-Lite: a Grammar of Interactive Multi-View Graphics

Indexed Chart



Focus+Context





Cross-filtering

Vega-Lite: a Grammar of Interactive 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

Abstract Data  Visual Representation

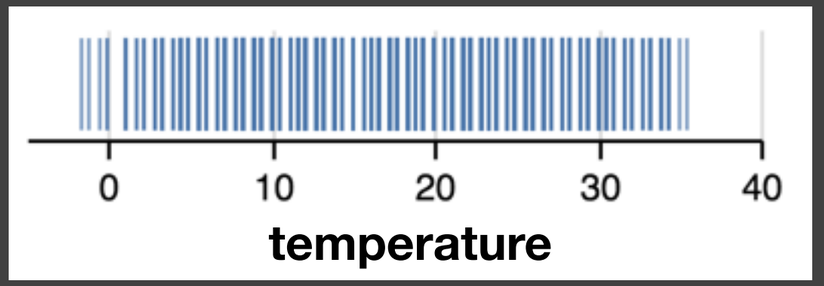
Specifying Single Views

Abstract Data → Visual Representation

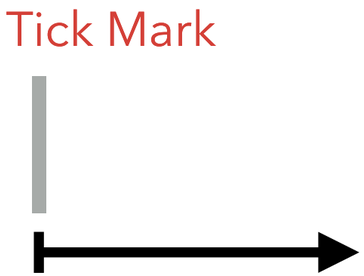
Weather Data for Seattle

date	temperature	precipitation	weather
1/1	10.6	10.9	"rain"
1/2	11.7	0.8	"drizzle"
1/3	12.2	10.2	"rain"
…	…	…	…

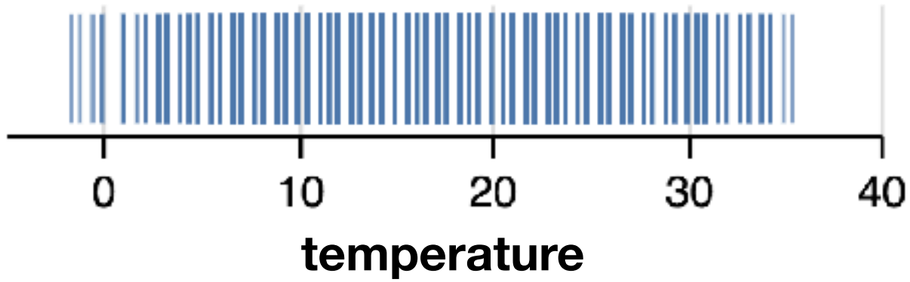
Strip Plot of Temperature



Strip Plot = (Tick with x=field)



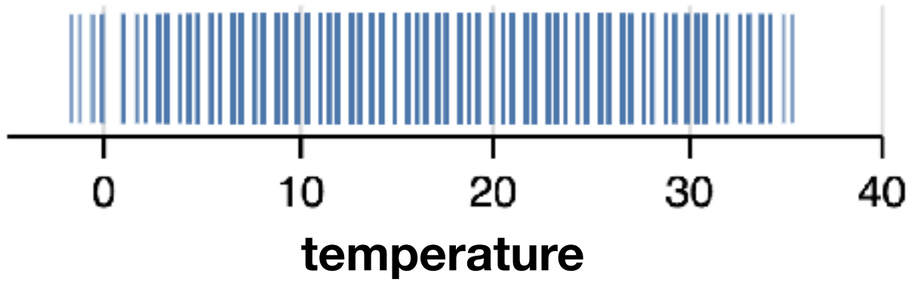

```
{
  data: {url: "weather-seattle.json"},
  mark: "tick",
  encoding: {
    x: {
      field: "temperature",
      type: "quantitative"
    }
  }
}
```



Vega-Lite is portable JSON specification

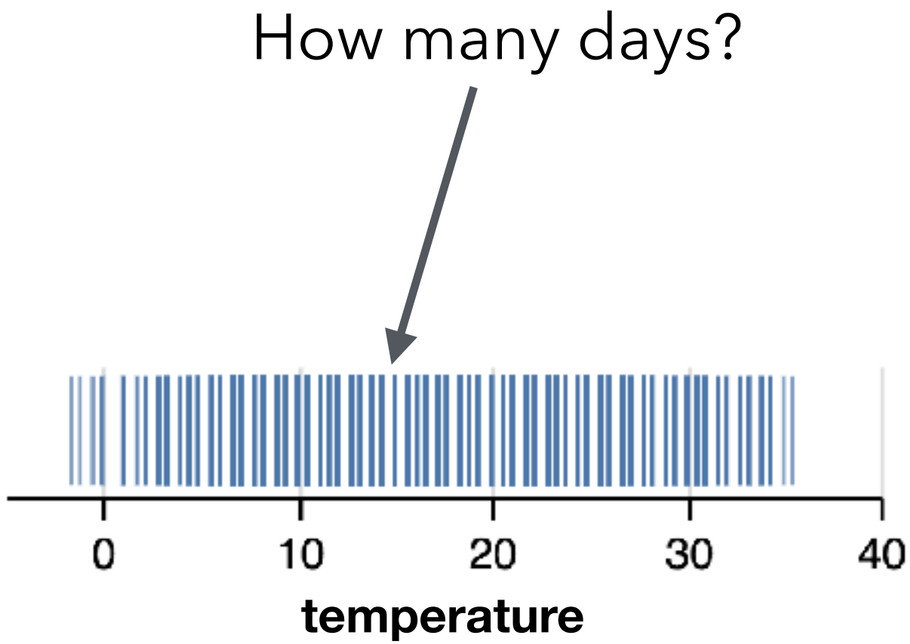
Strip Plot: Default Scales and Axes

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



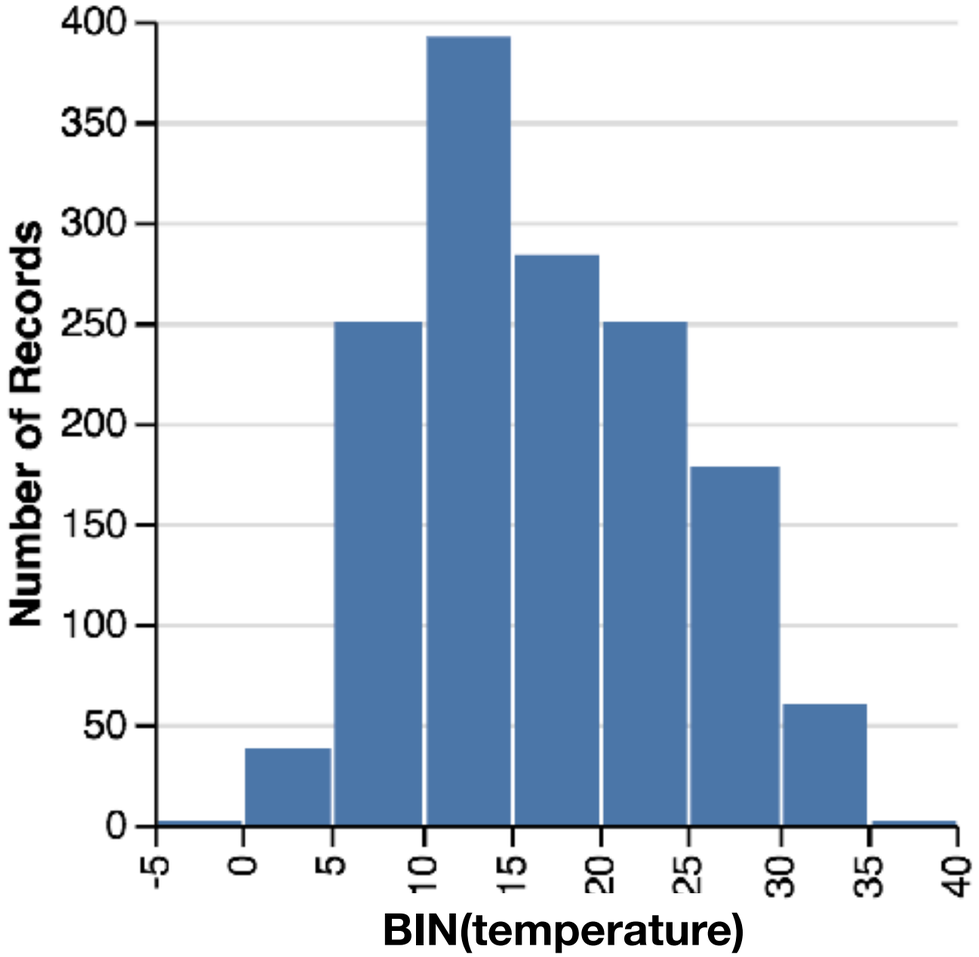
Strip Plot

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



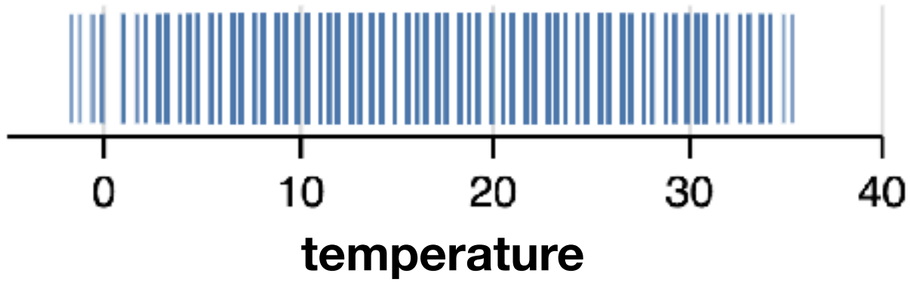
Histogram = (Bar with x=binned field, y=count)

Goal



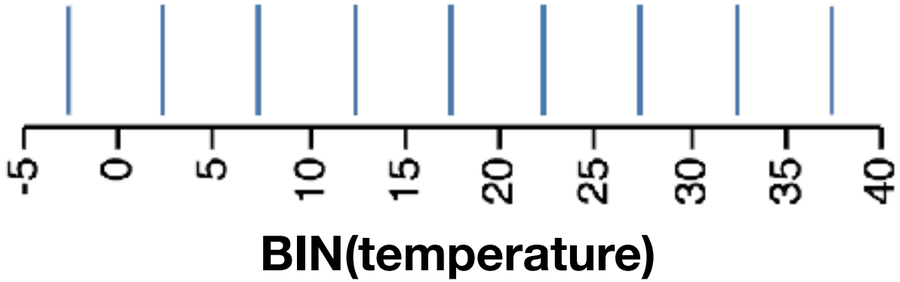
Histogram = (Bar with x=binned field, y=count)

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



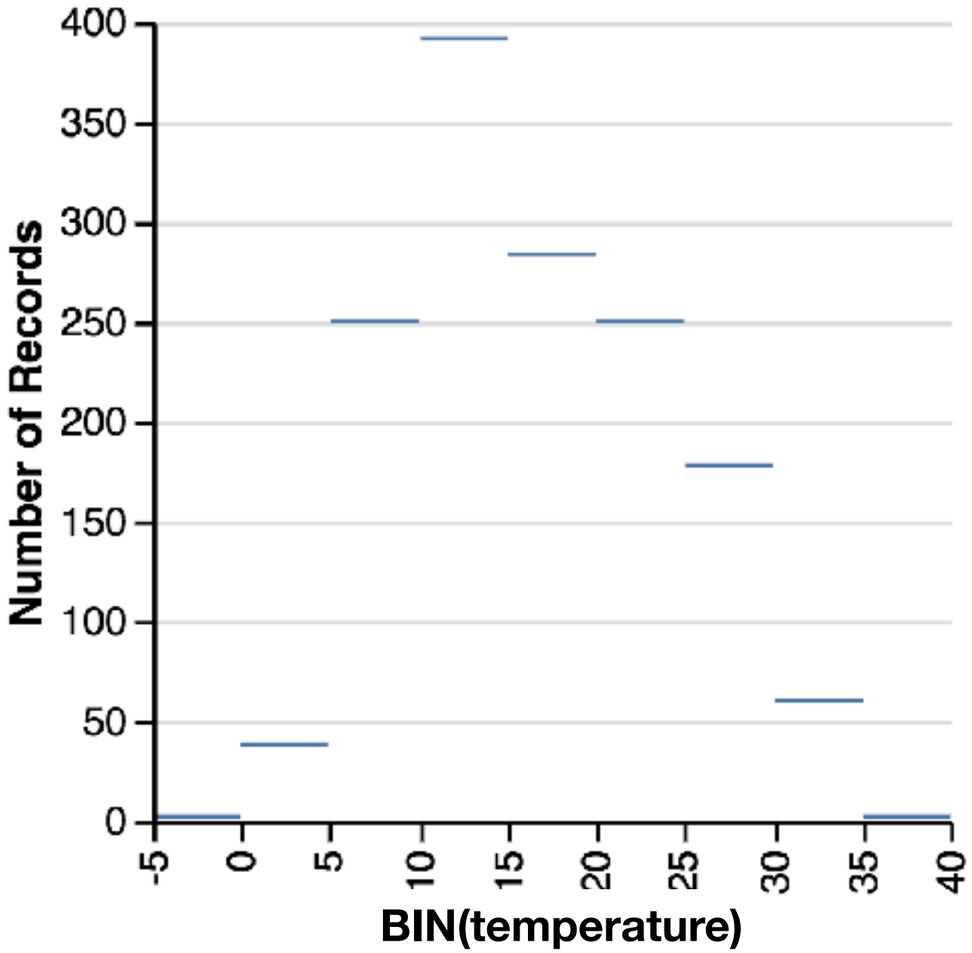
Histogram = (Bar with x=binned field, y=count)

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



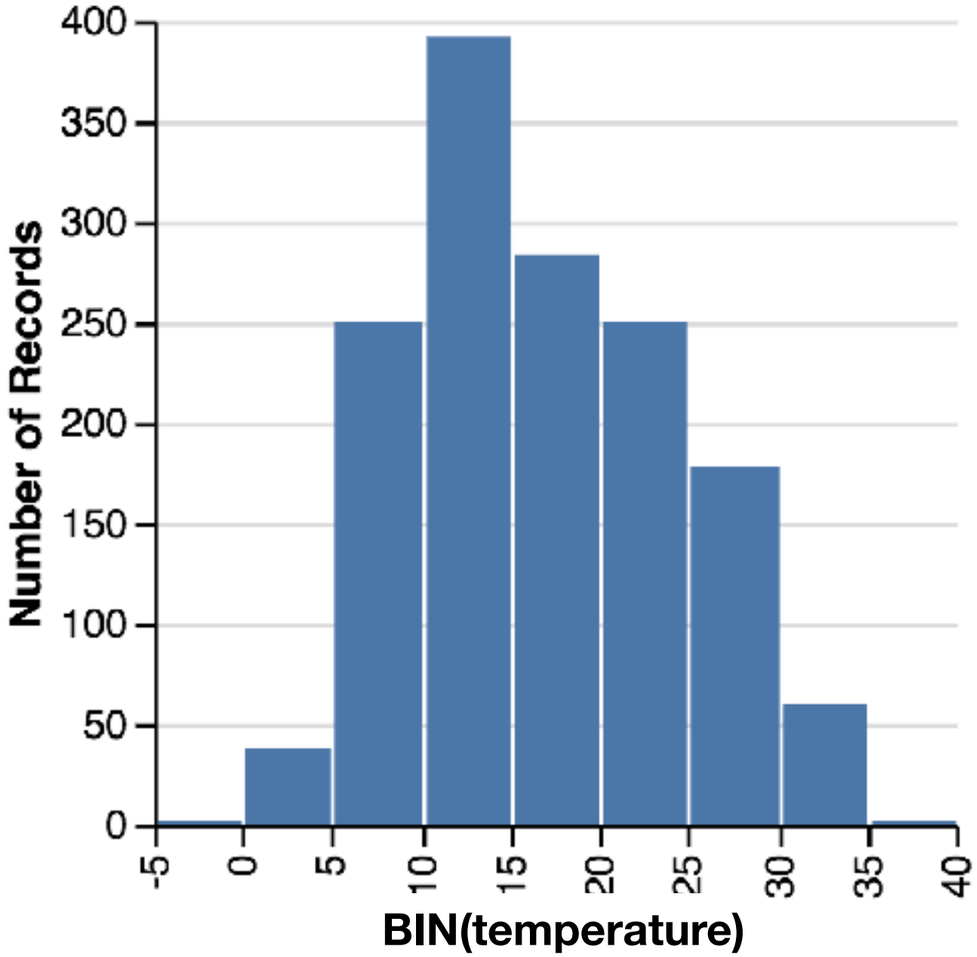
Histogram = (Bar with x=binned field, y=count)

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



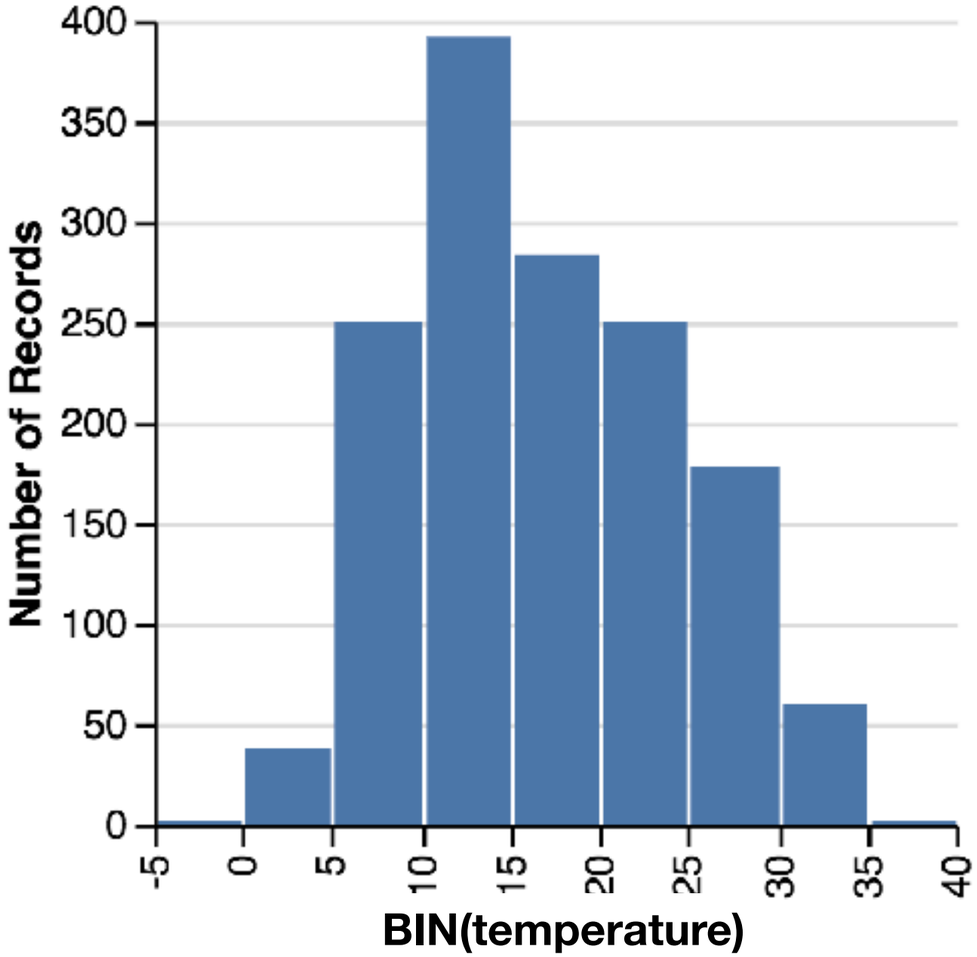
Histogram = (Bar with x=binned field, y=count)

```
{
  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=count)

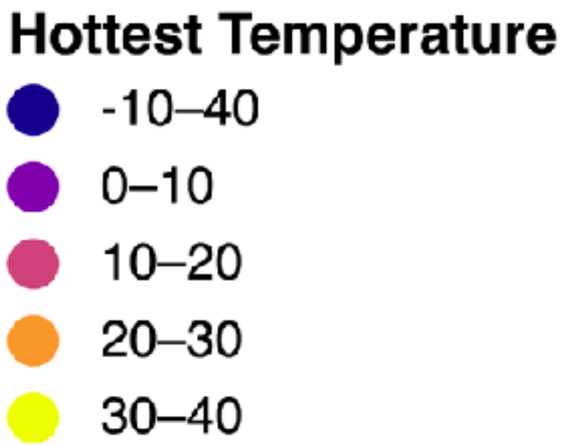
```
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
  }
}
```

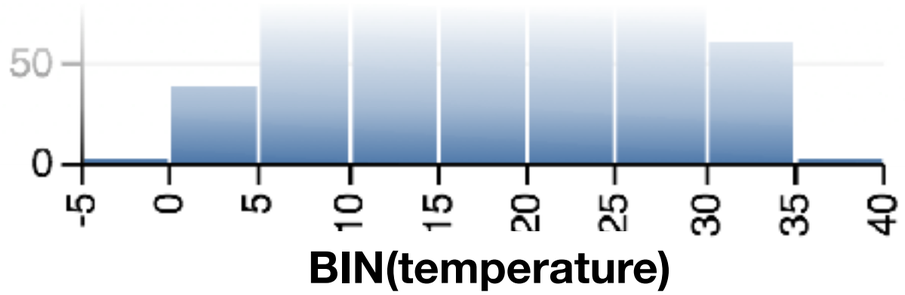


Sensible Defaults for Binning

Channel determines guide and bin parameters

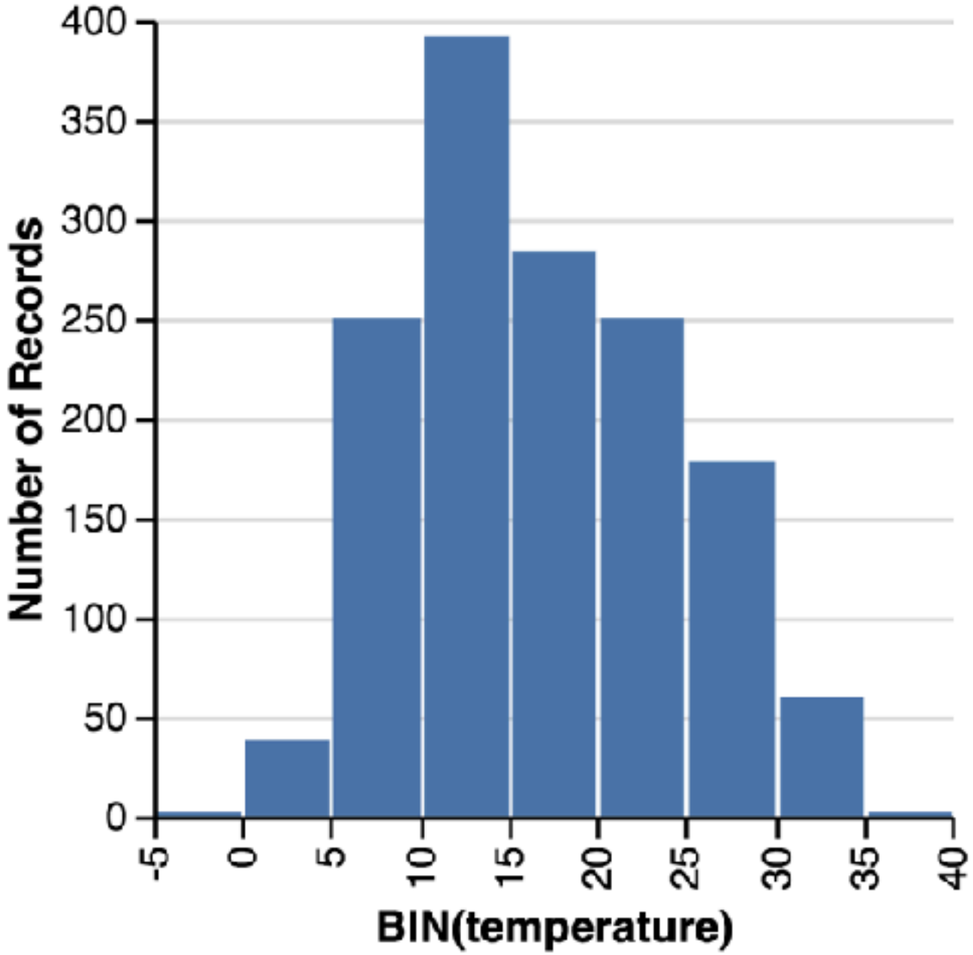
	Color/Opacity/Shape	Position
Guide	Legend with range labels	Quantitative axis
# of Bins	Fewer bins	More bins





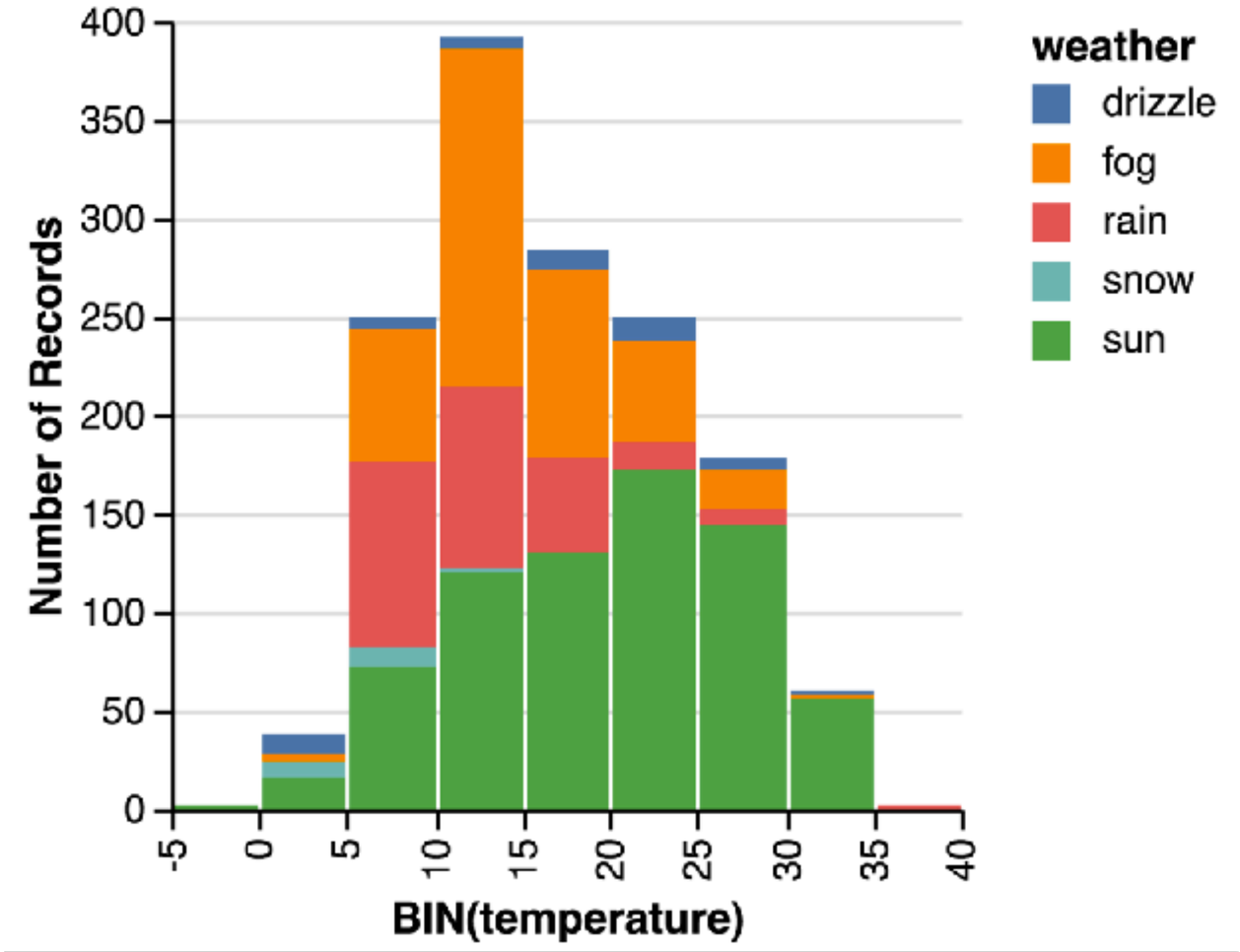
Histogram

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



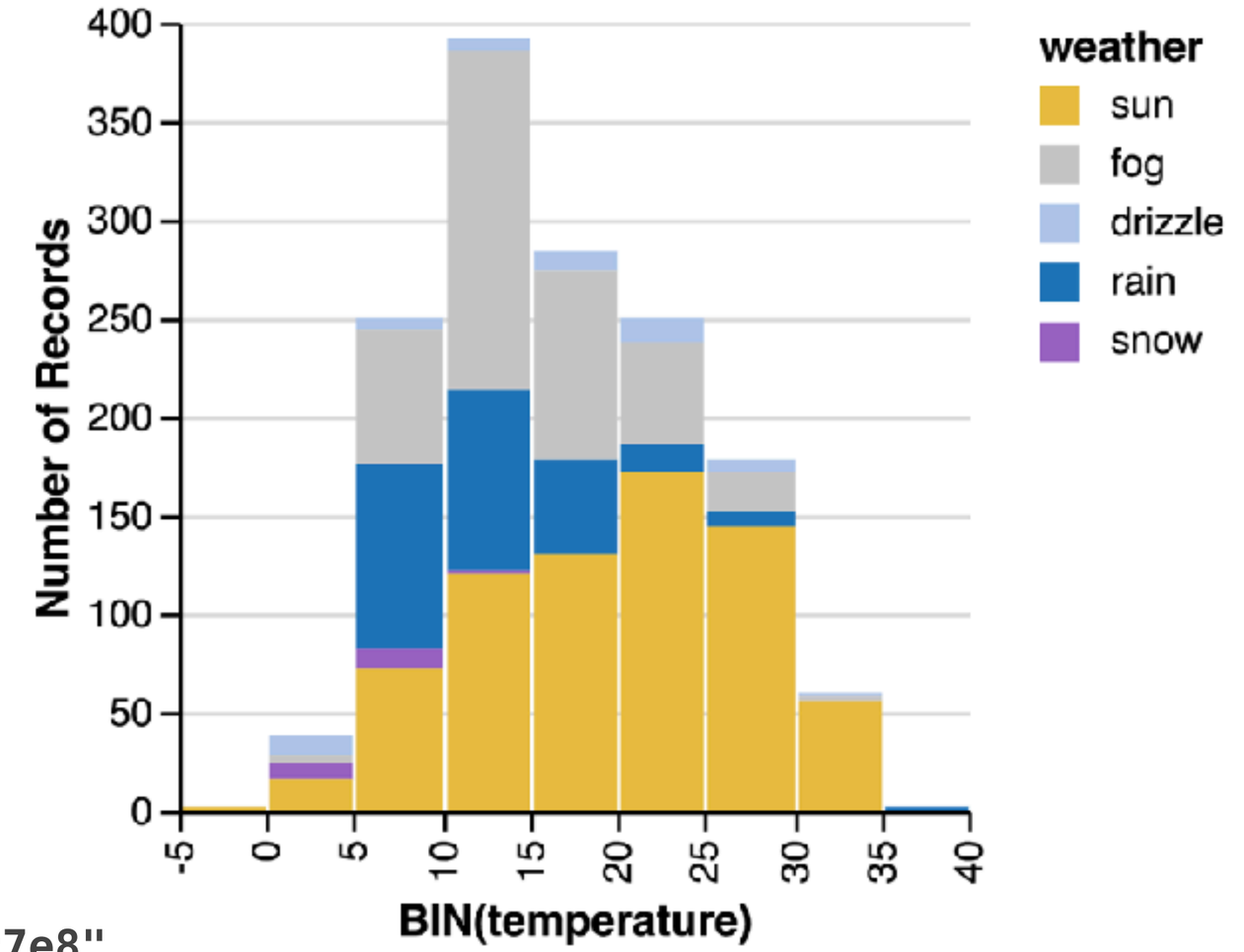
Histogram + Color

```
{
  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"
    }
  }
}
```



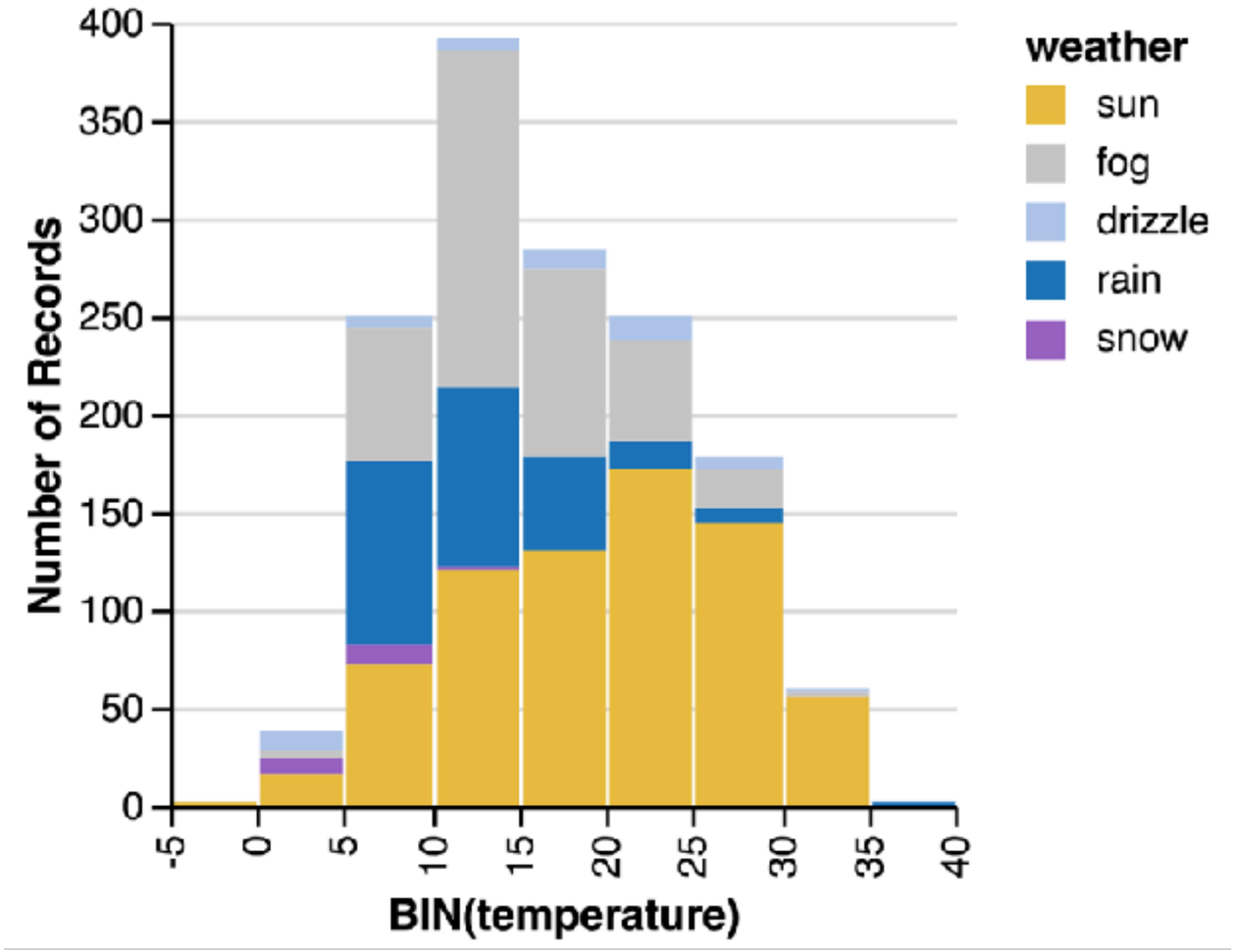
Histogram + Color

```
{
  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

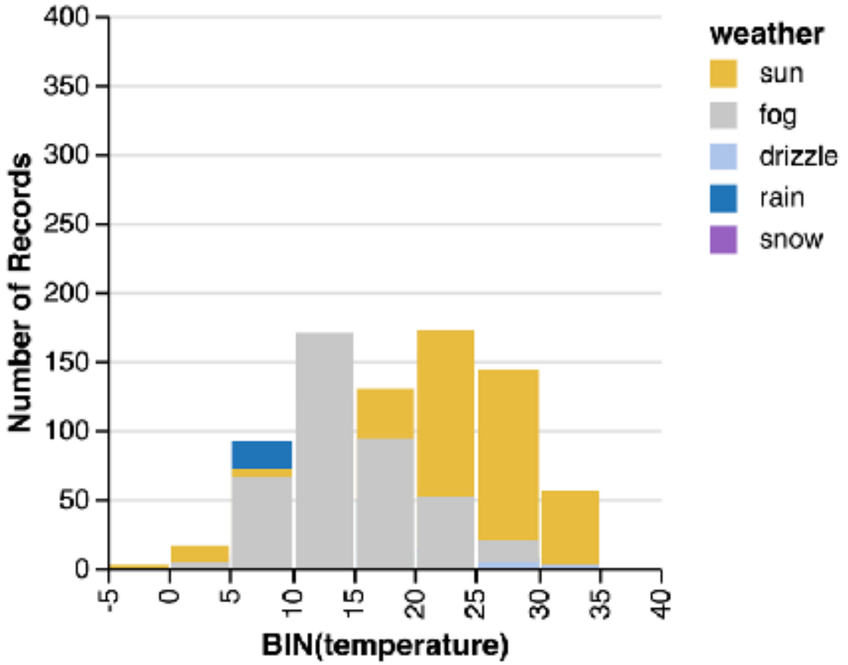
```
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    },
    color: {
      field: "weather",
      type: "nominal",
      ...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



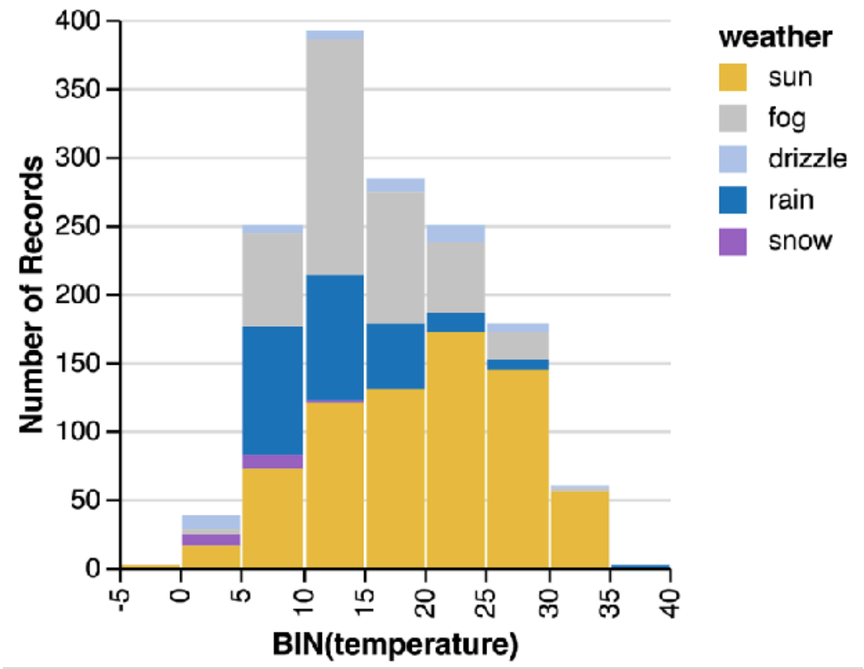
Stacked Histogram: Sensible Defaults

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

no stack



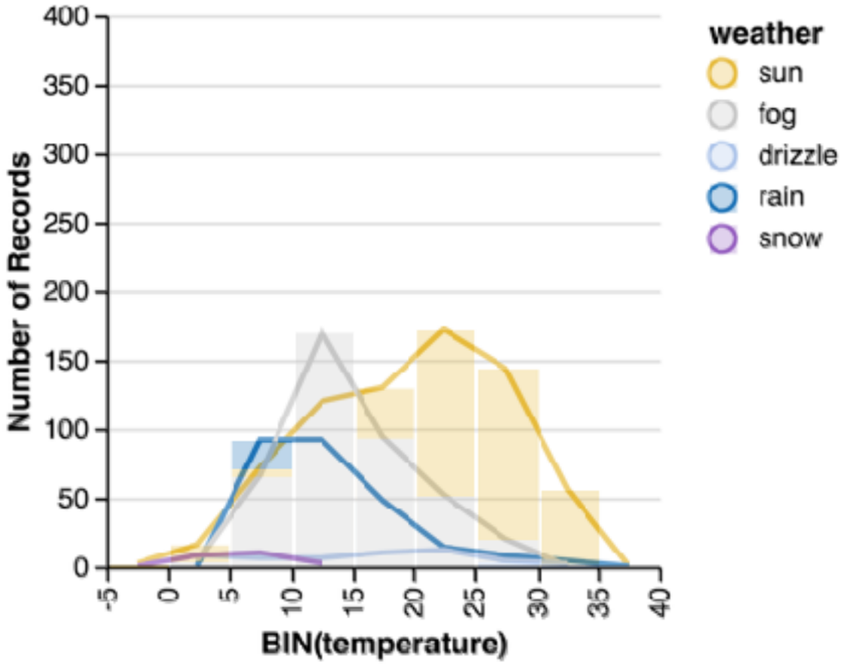
stack (default)



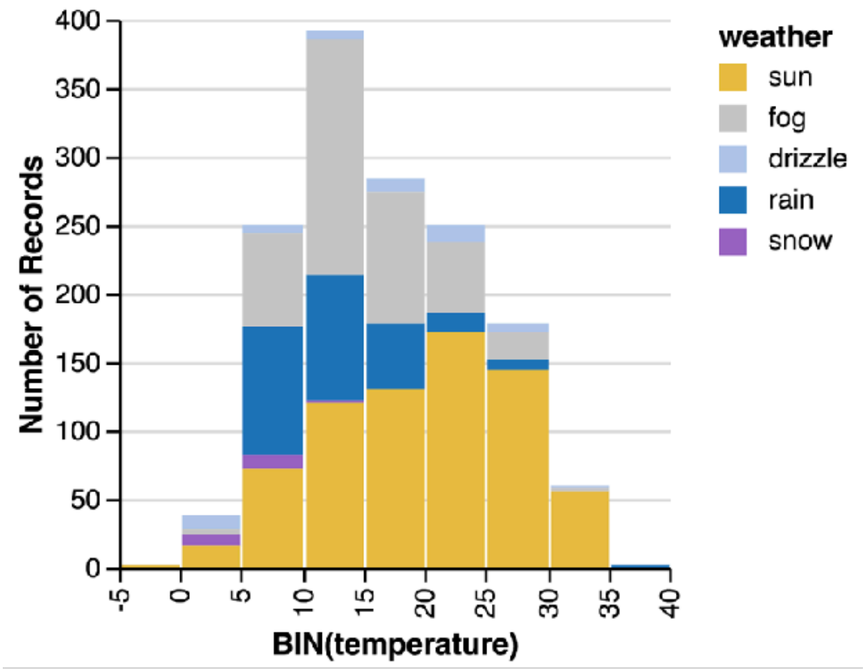
Stacked Histogram: Sensible Defaults

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

no stack → overlap

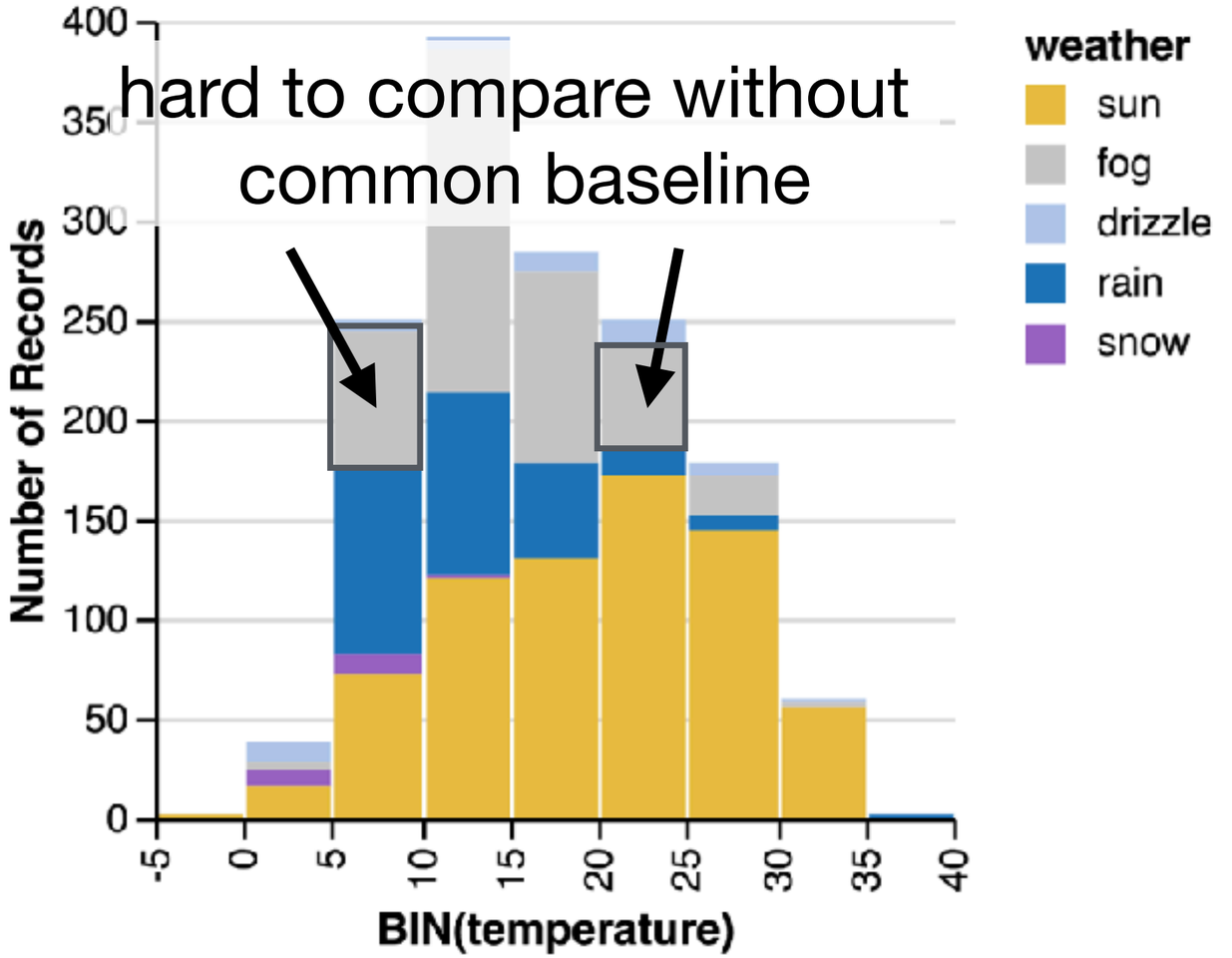


stack (default)



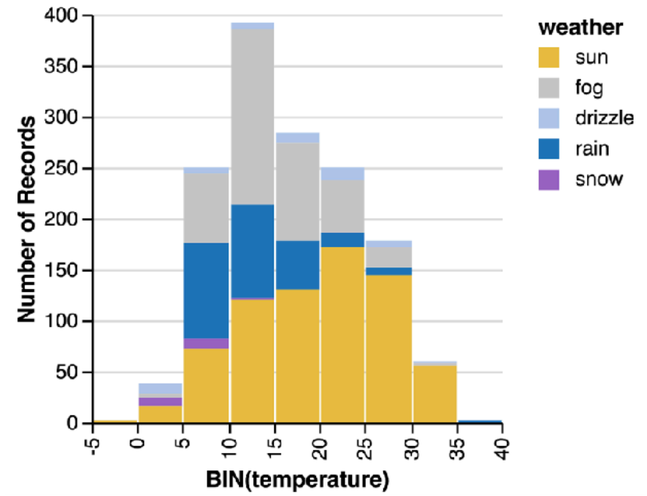
Histogram + Color = Stacked Histogram

```
{
  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"
    }
  }
}
```



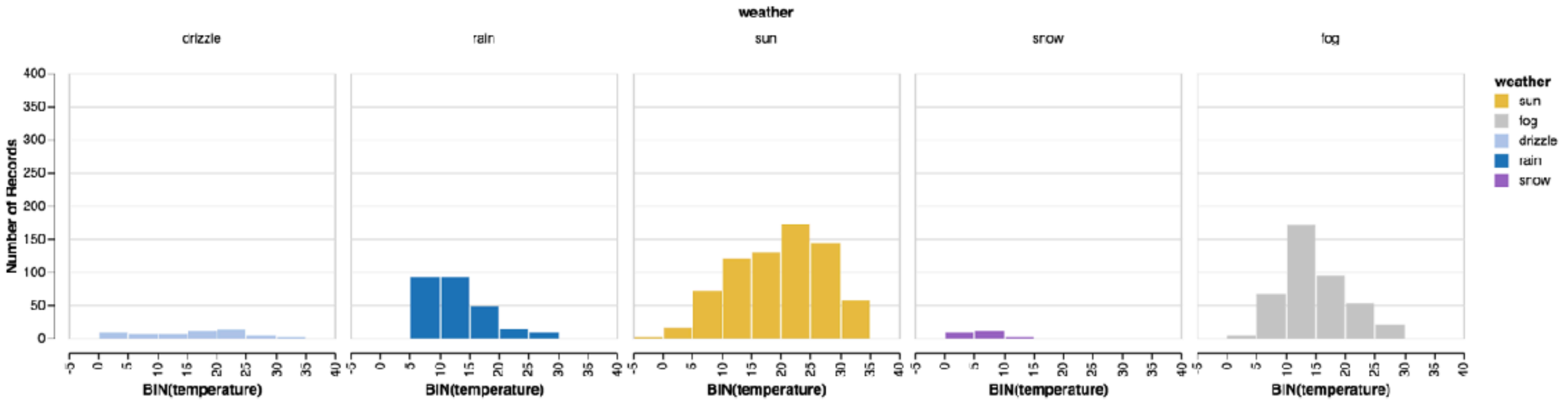
Histogram + Color = Stacked Histogram

```
{
  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

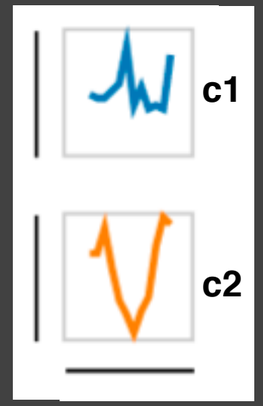
Using Vega-Lite

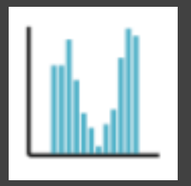
- **Programming** with Vega-Lite
- Higher-level **Tools** and **Recommendations**

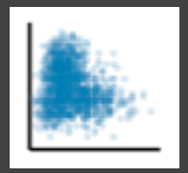
View Composition Operators

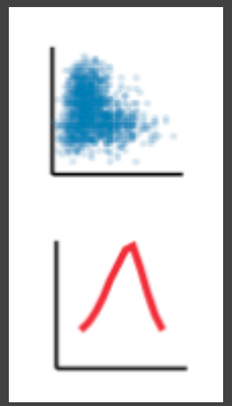
facet row: C

=

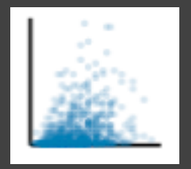


layer: [ , ] =

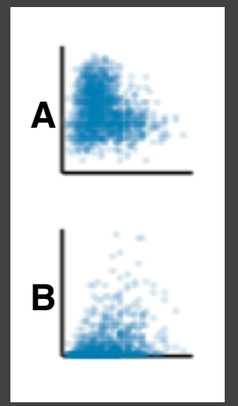
vconcat: [ , ] =



repeat row: [A,B]

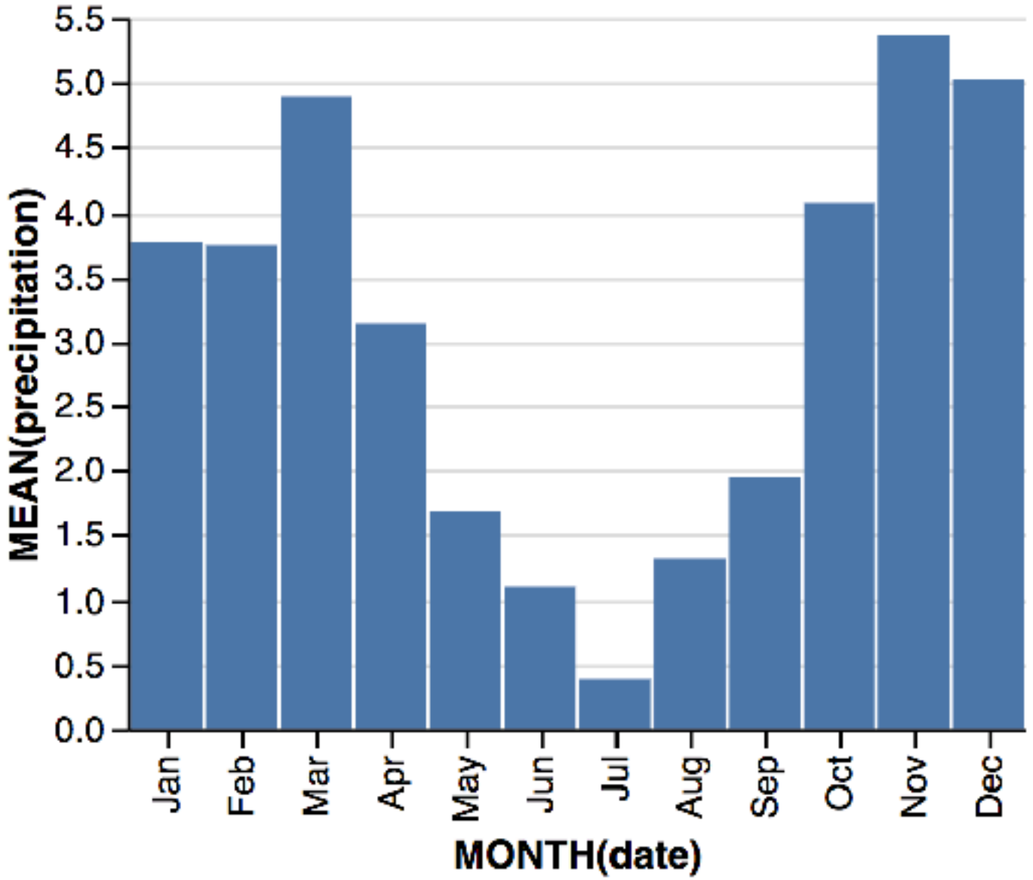


=



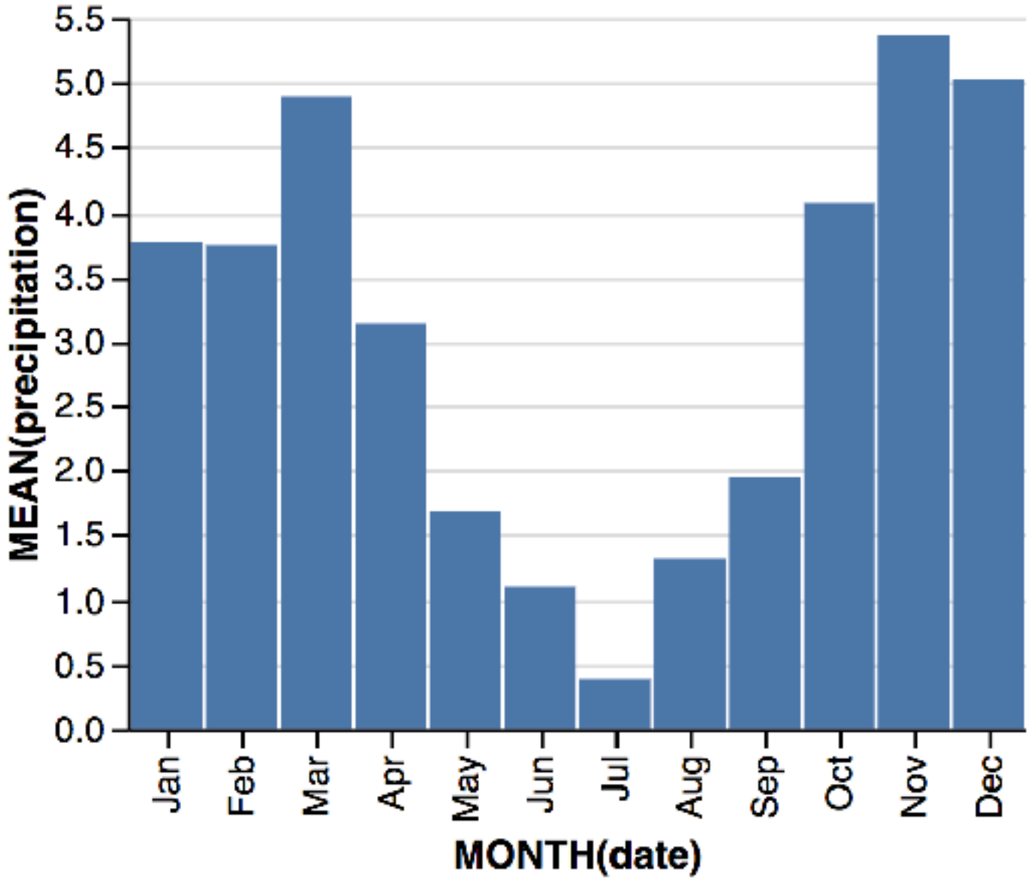
Monthly Precipitation

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



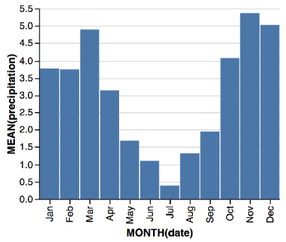
Layering

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

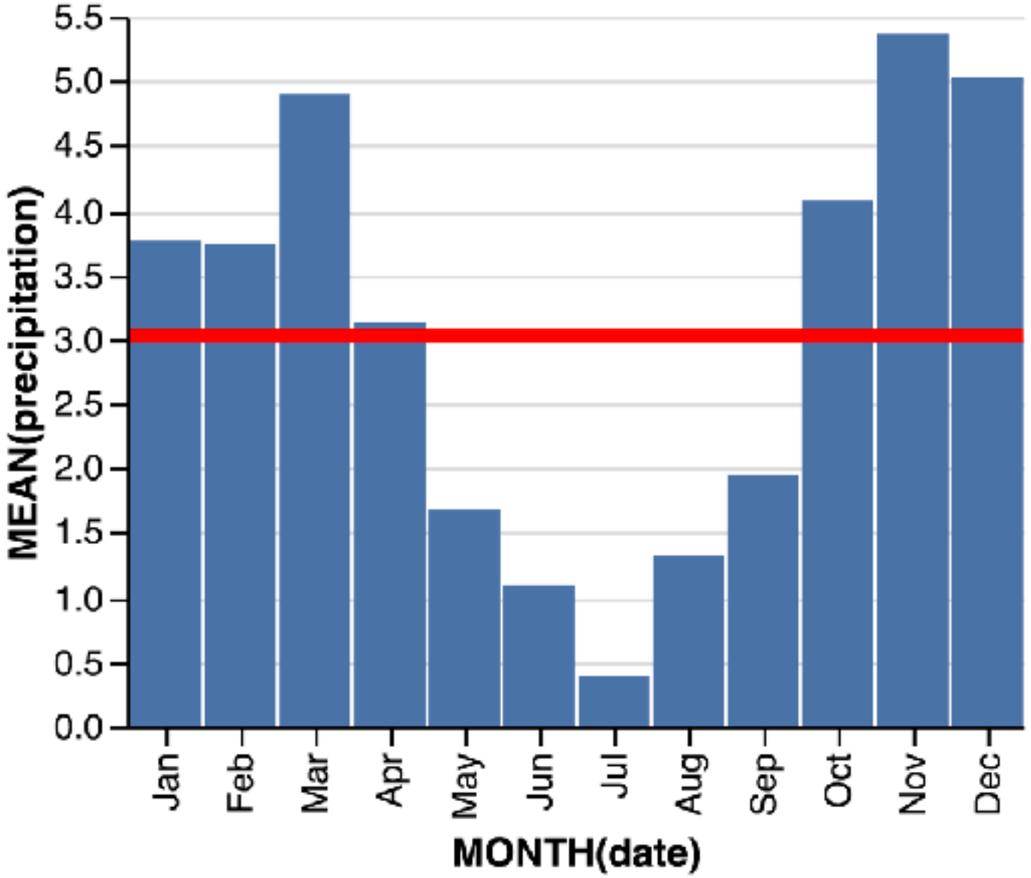


Layering

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

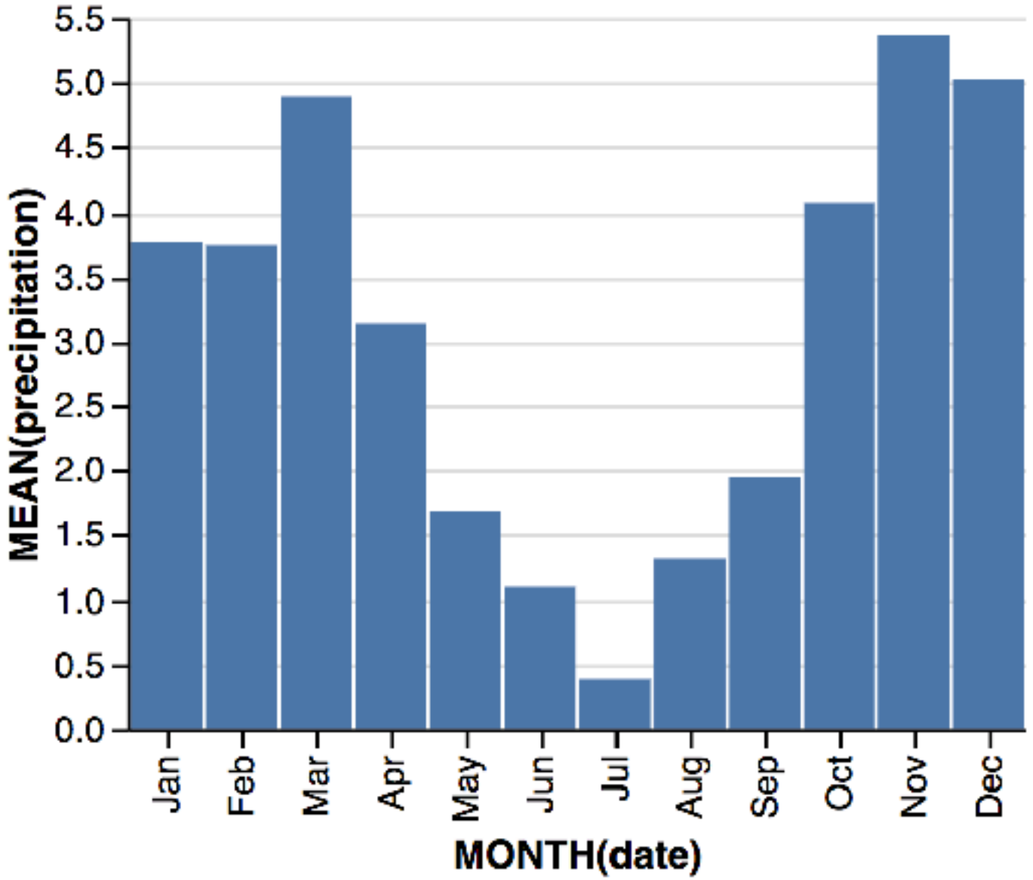



```
  }, {
    mark: "rule",
    encoding: {
      y: {
        aggregate: "mean",
        field: "precipitation",
        type: "quantitative"
      }
    }]
}
```



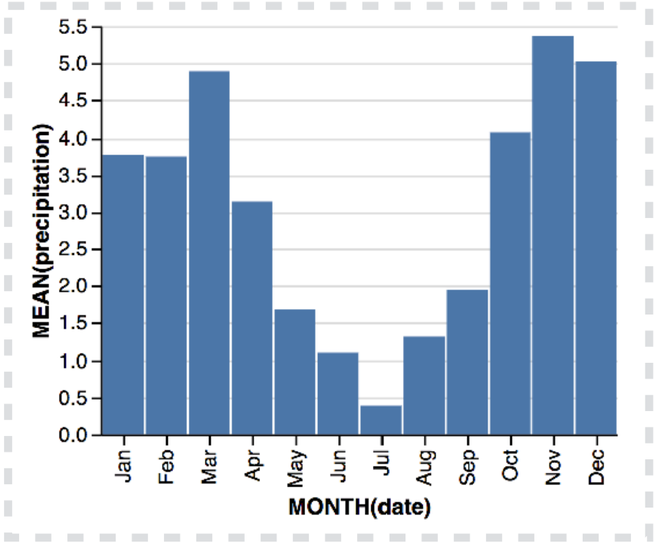
Concat

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



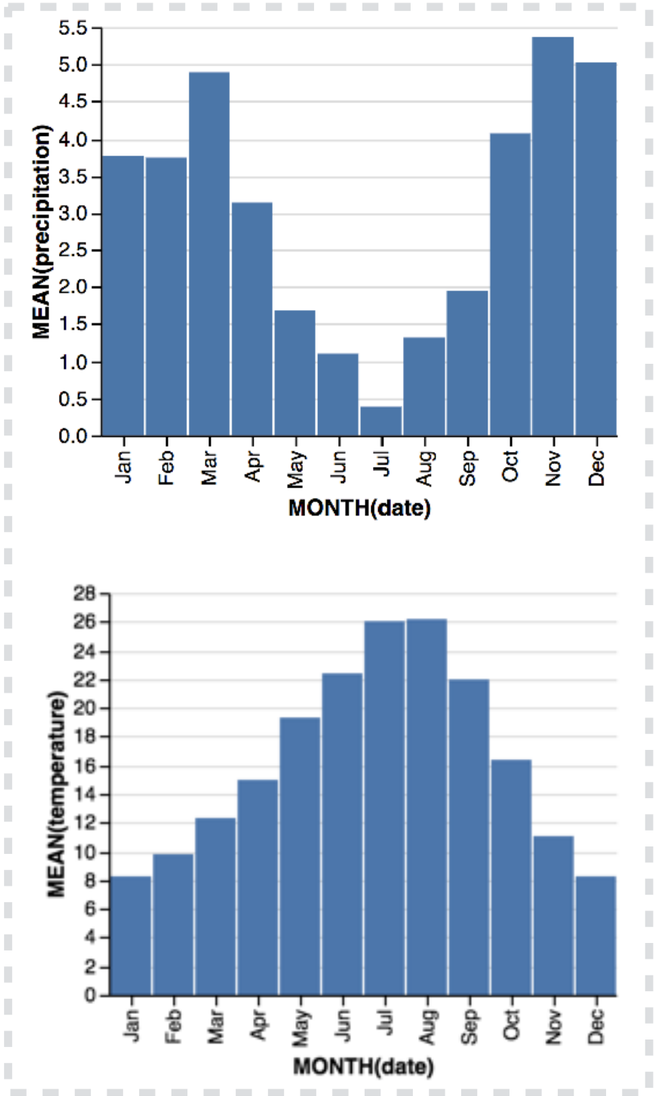
Concat

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



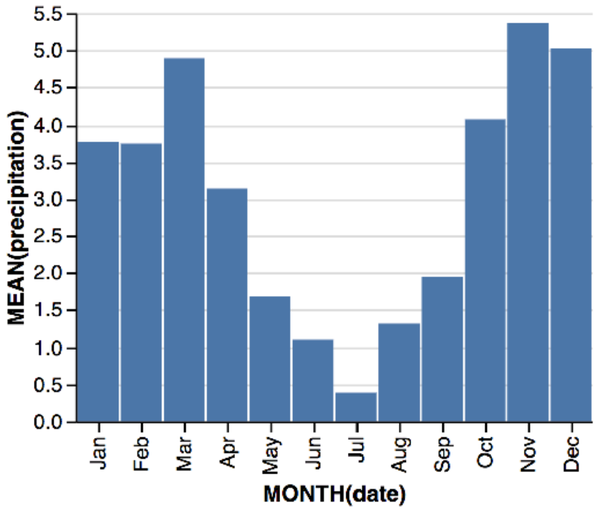
Concat

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



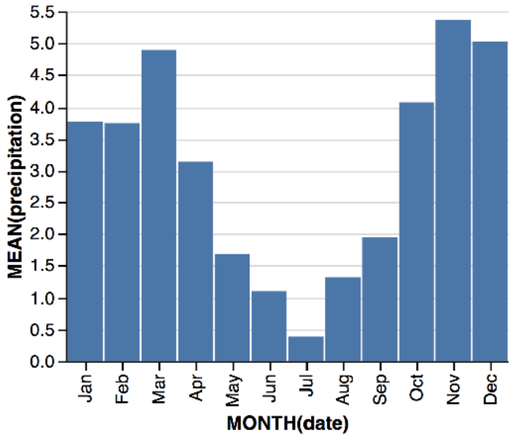
Repeat

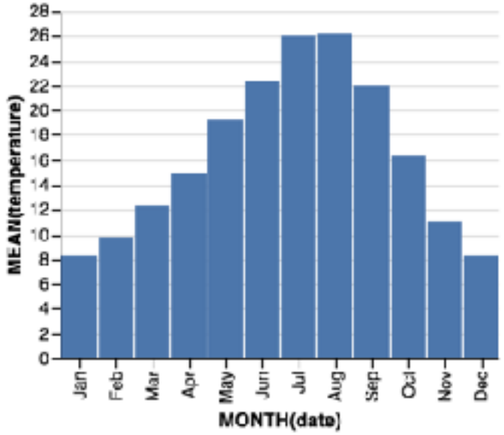
```
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      timeUnit: "month",
      field: "date",
      type: "ordinal"
    },
    y: {
      aggregate: "mean",
      field: "precipitation",
      type: "quantitative"
    }
  }
}
```

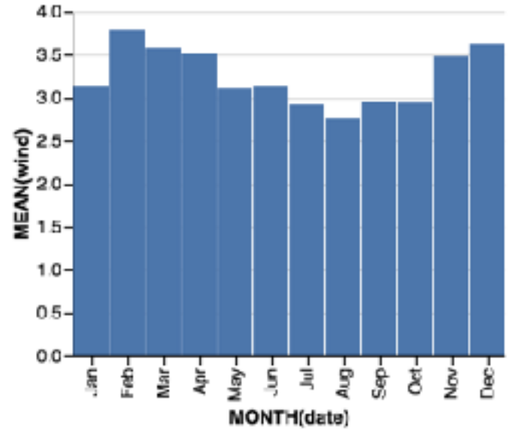


Repeat

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

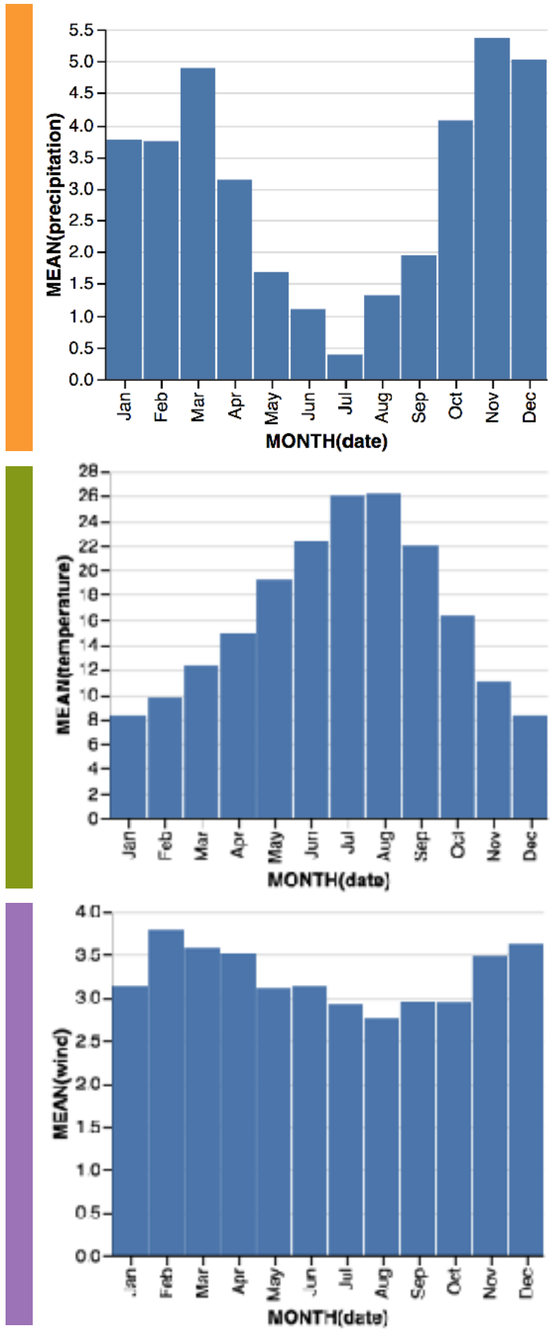




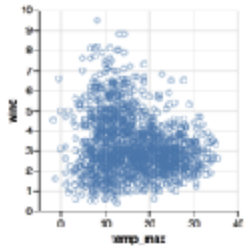


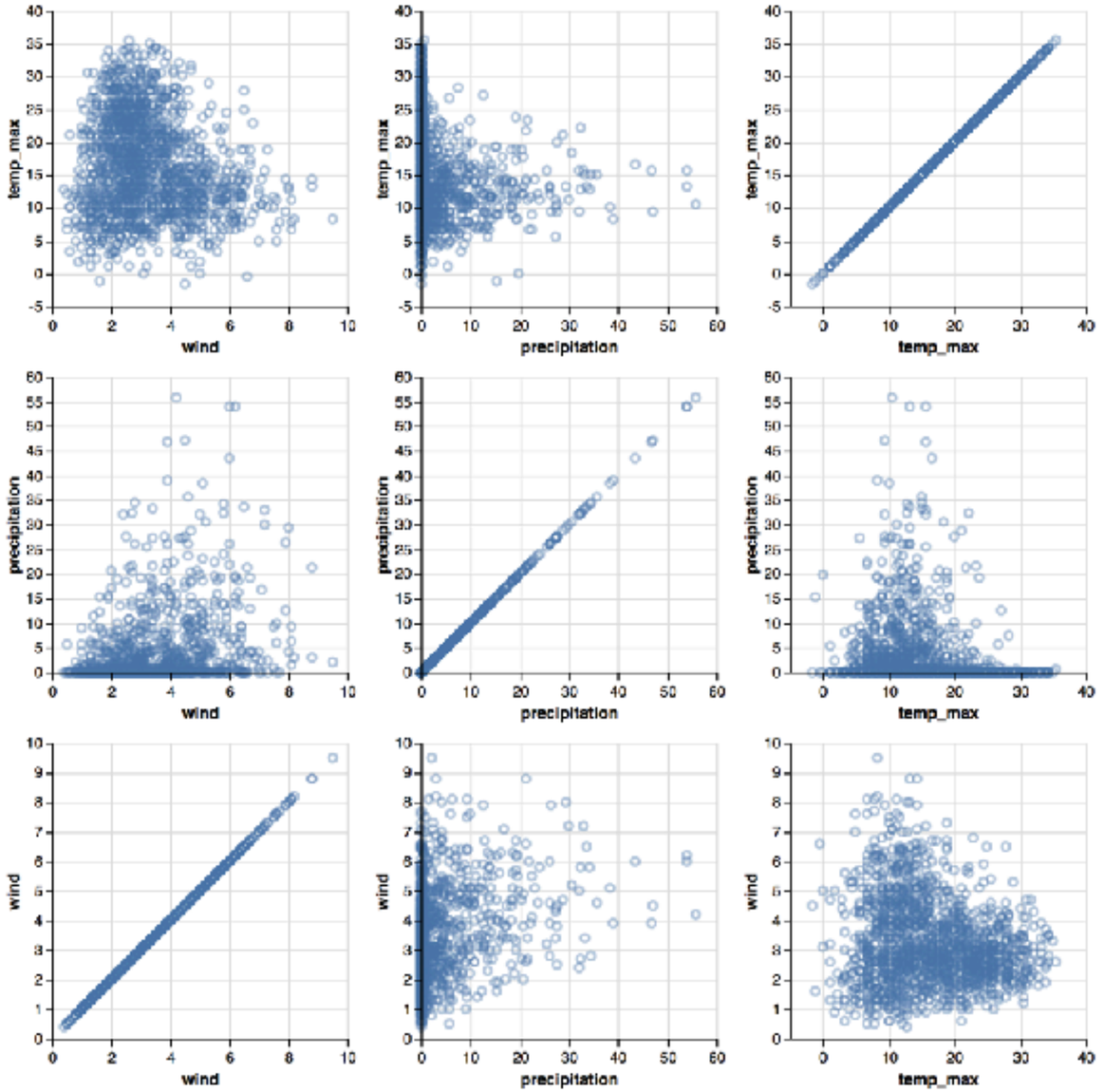
Repeat

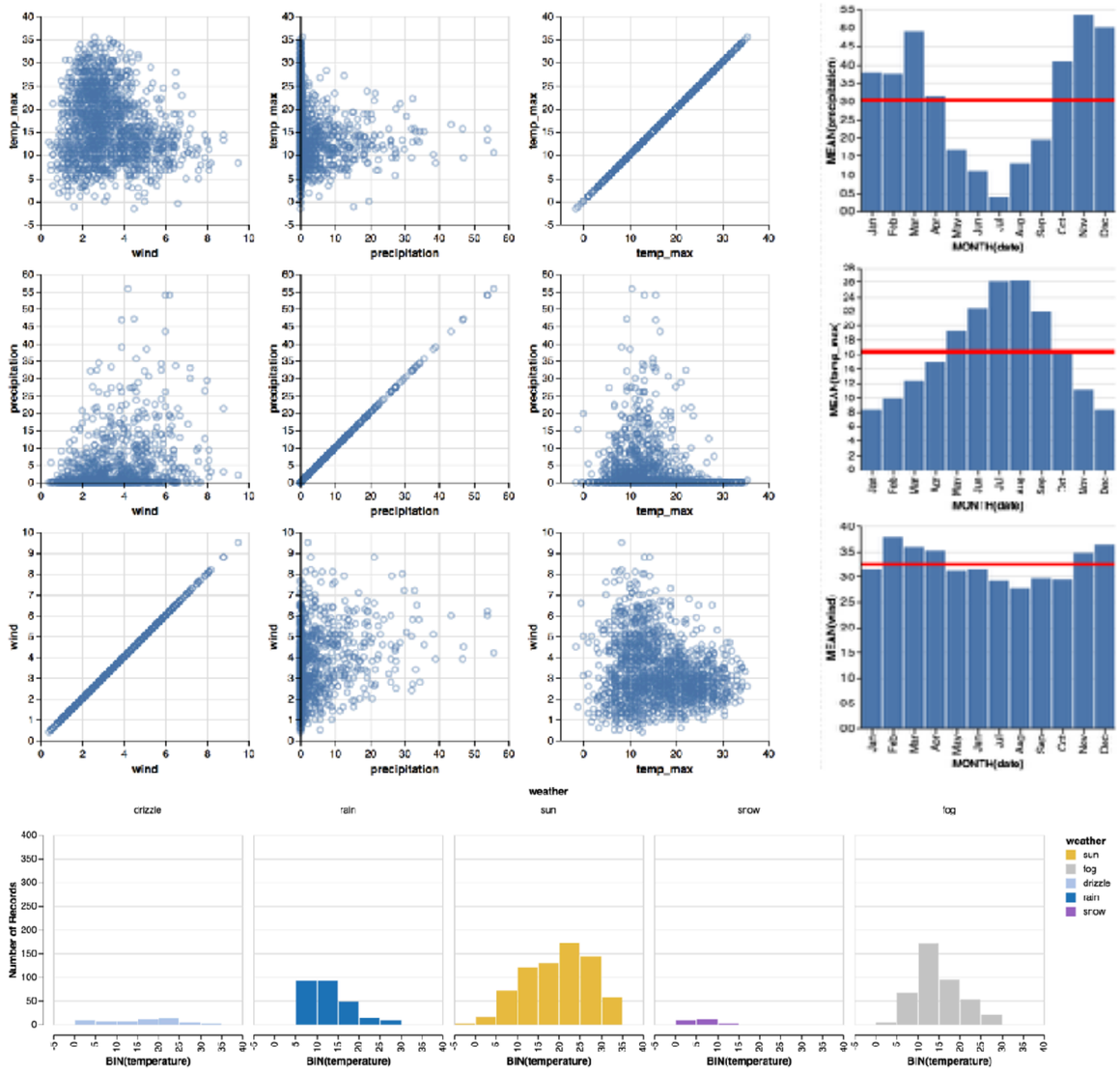
```
{
  repeat: {
    column: ["precipitation", "temperature", "wind"]
  },
  spec: {
    data: {url: "weather-seattle.json"},
    mark: "bar",
    encoding: {
      x: {
        timeUnit: "month",
        field: "date",
        type: "ordinal"
      },
      y: {
        aggregate: "mean",
        field: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



Repeat: SPLOM

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



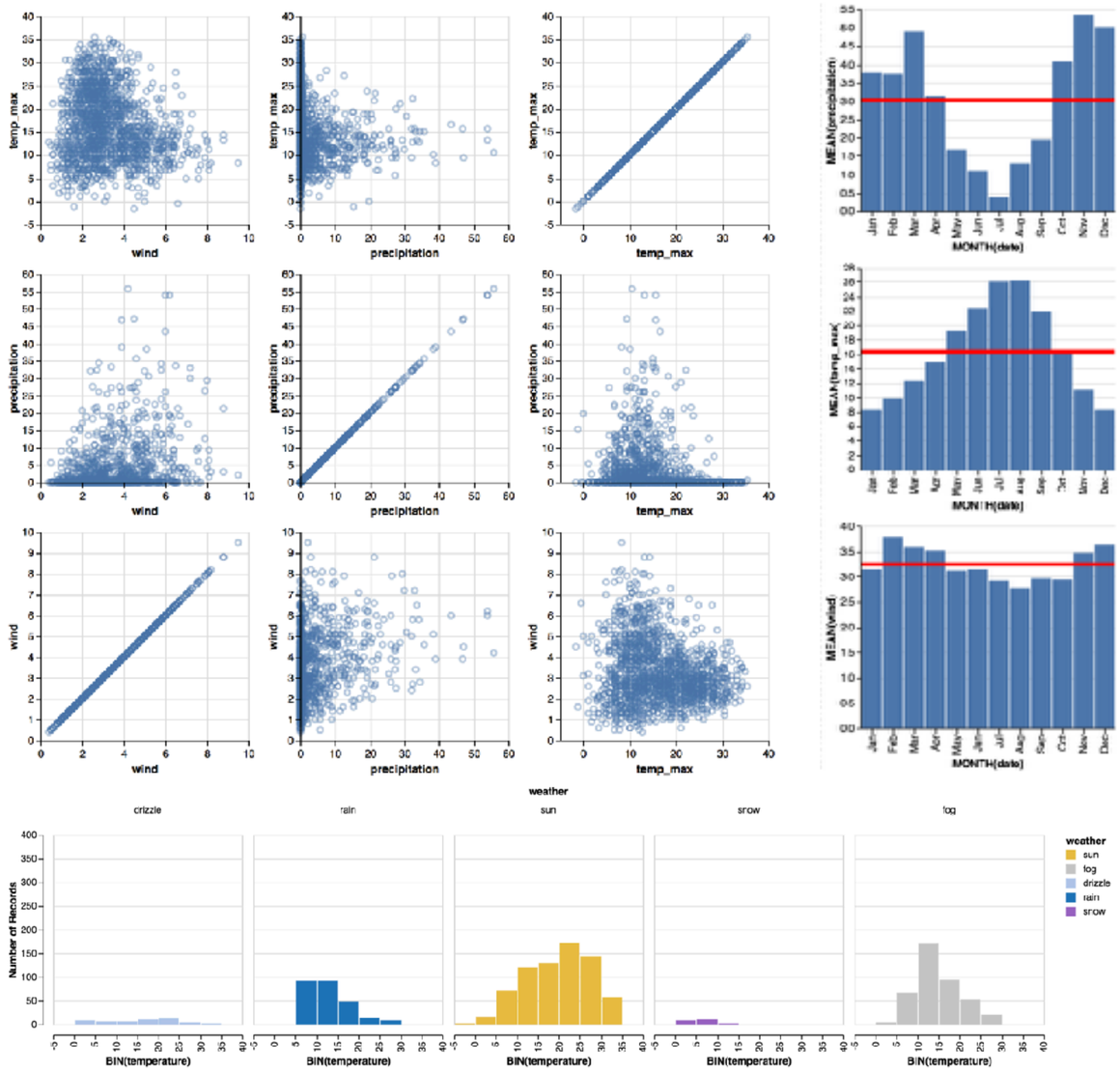


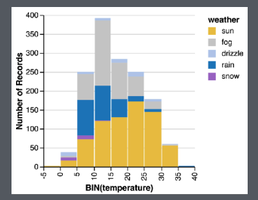
Dashboards

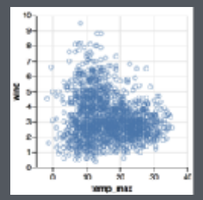
Many views, some of them composed

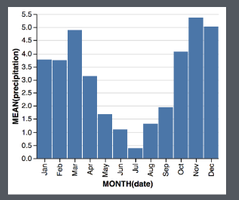
Manual data, and view management

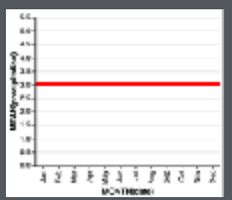
Composition



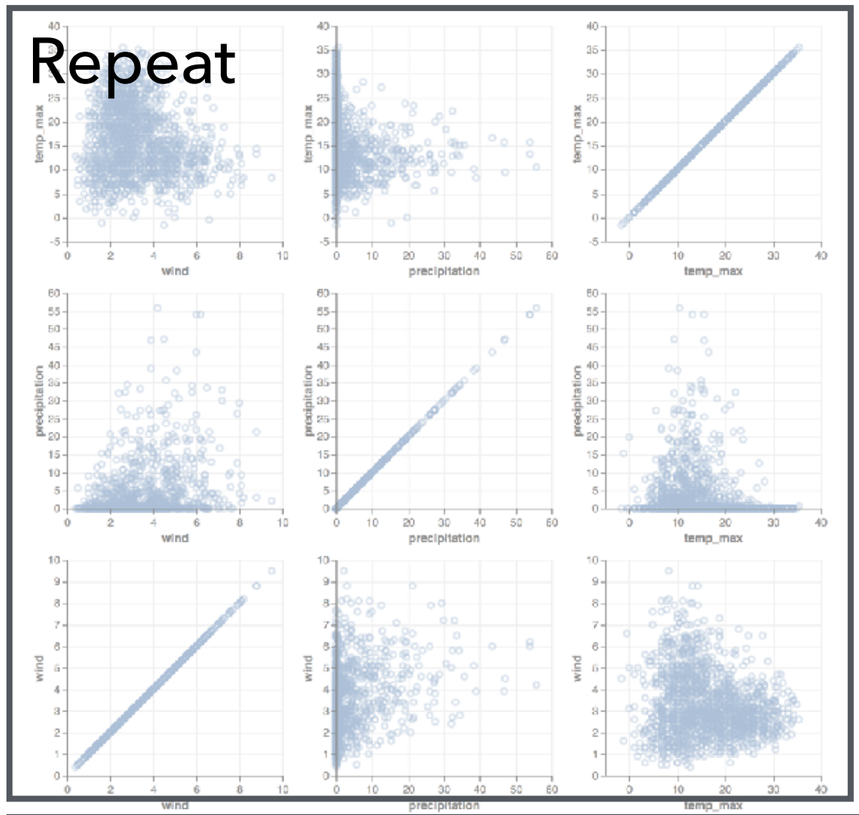


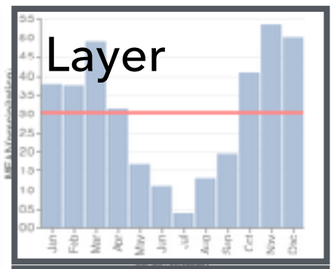


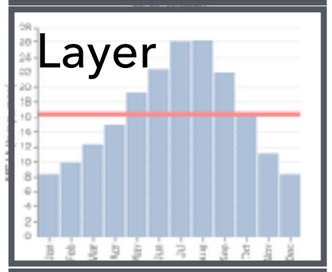


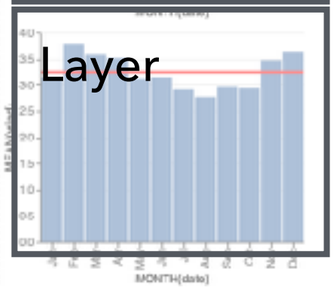


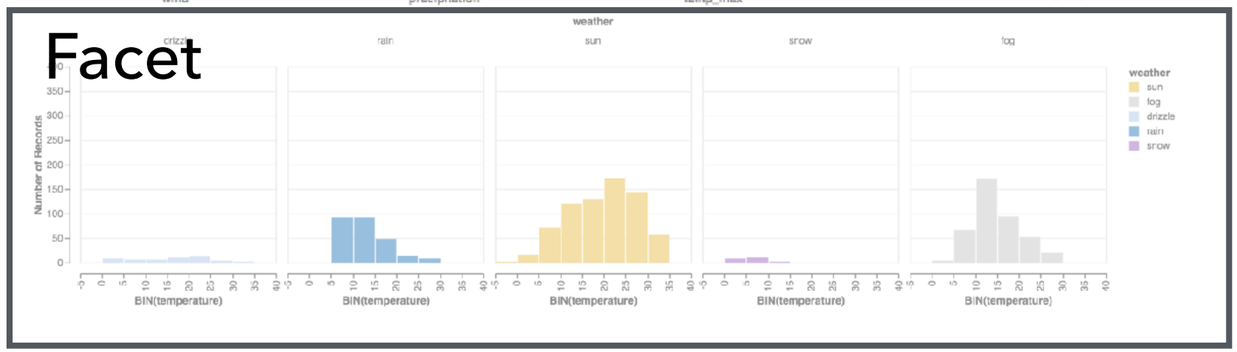
Composition

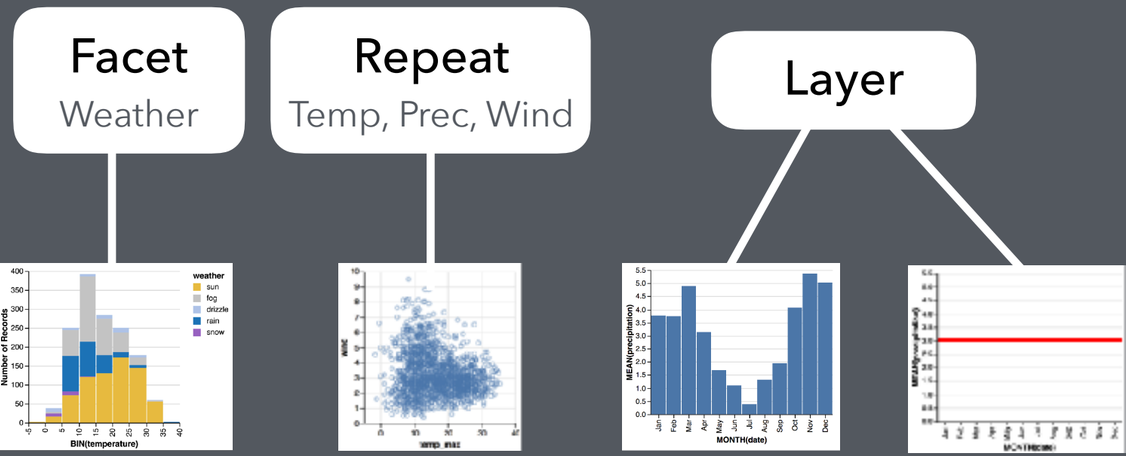




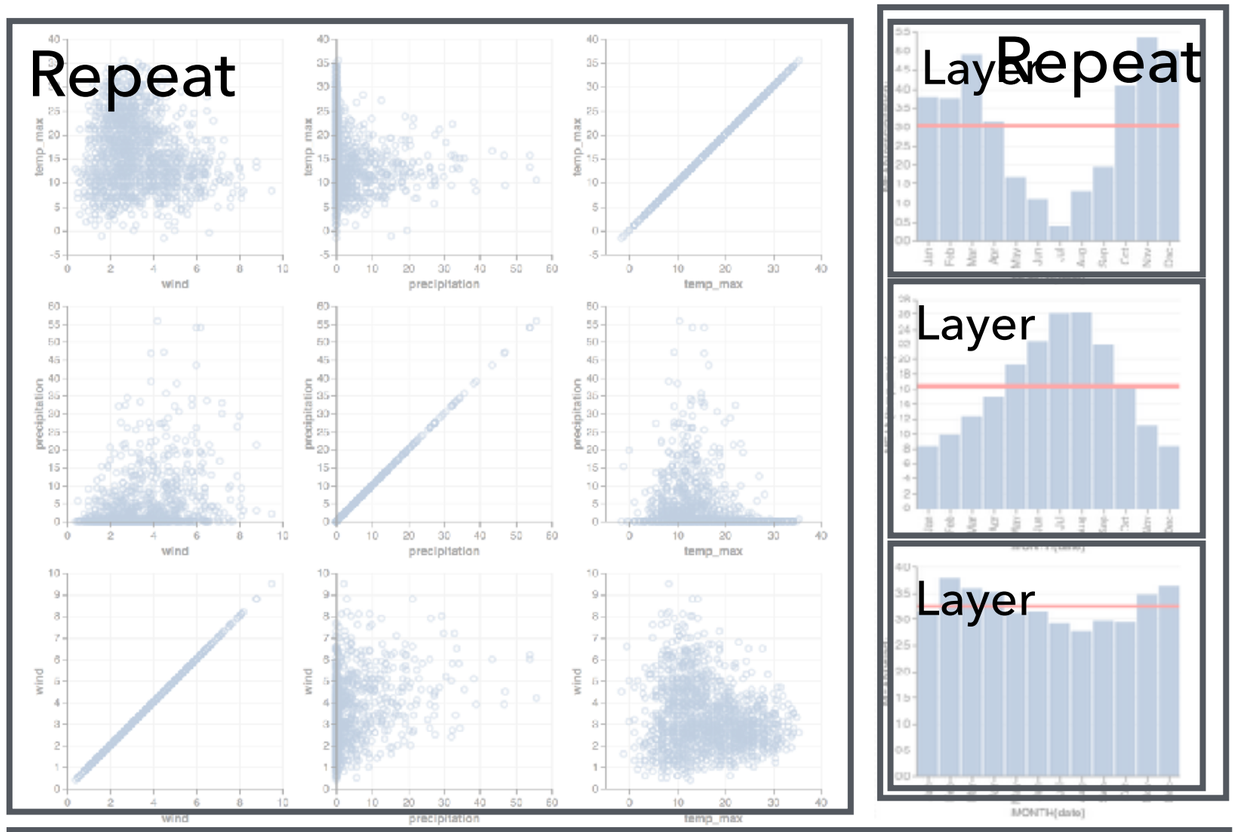


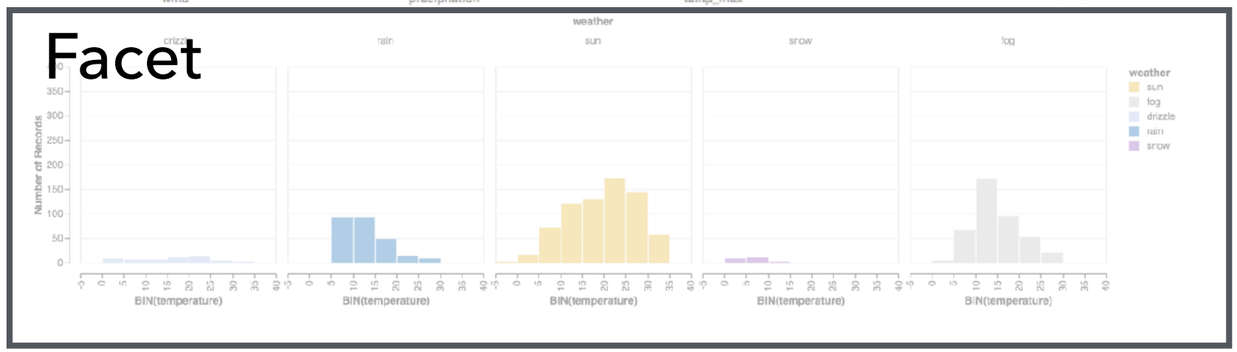


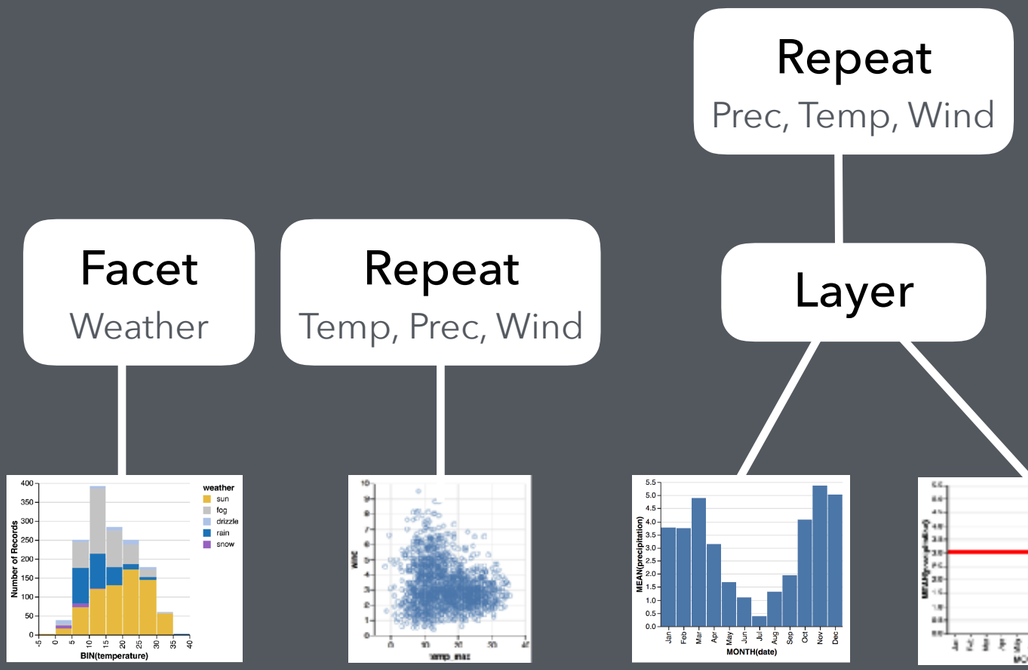




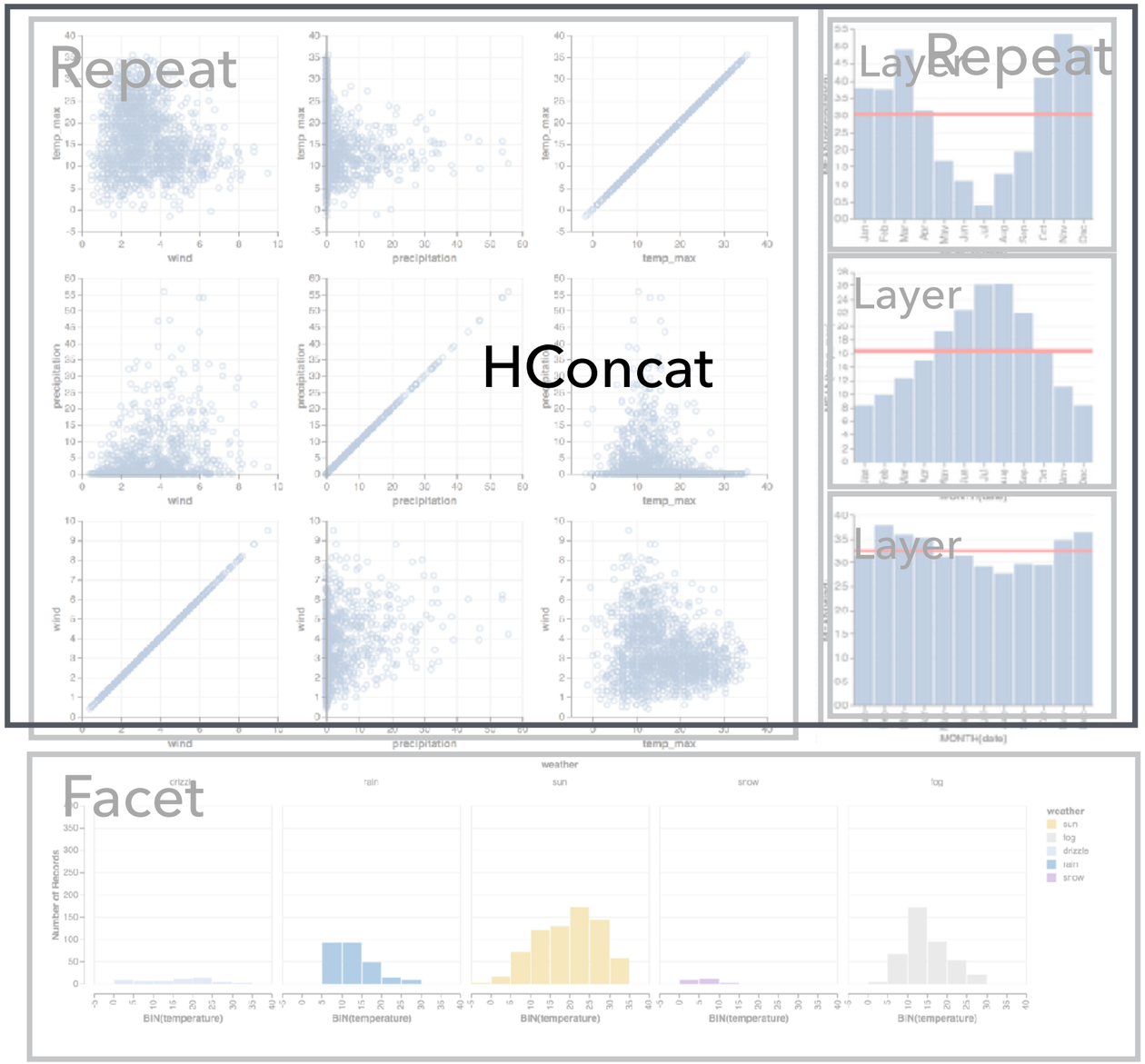
Hierarchical View Composition

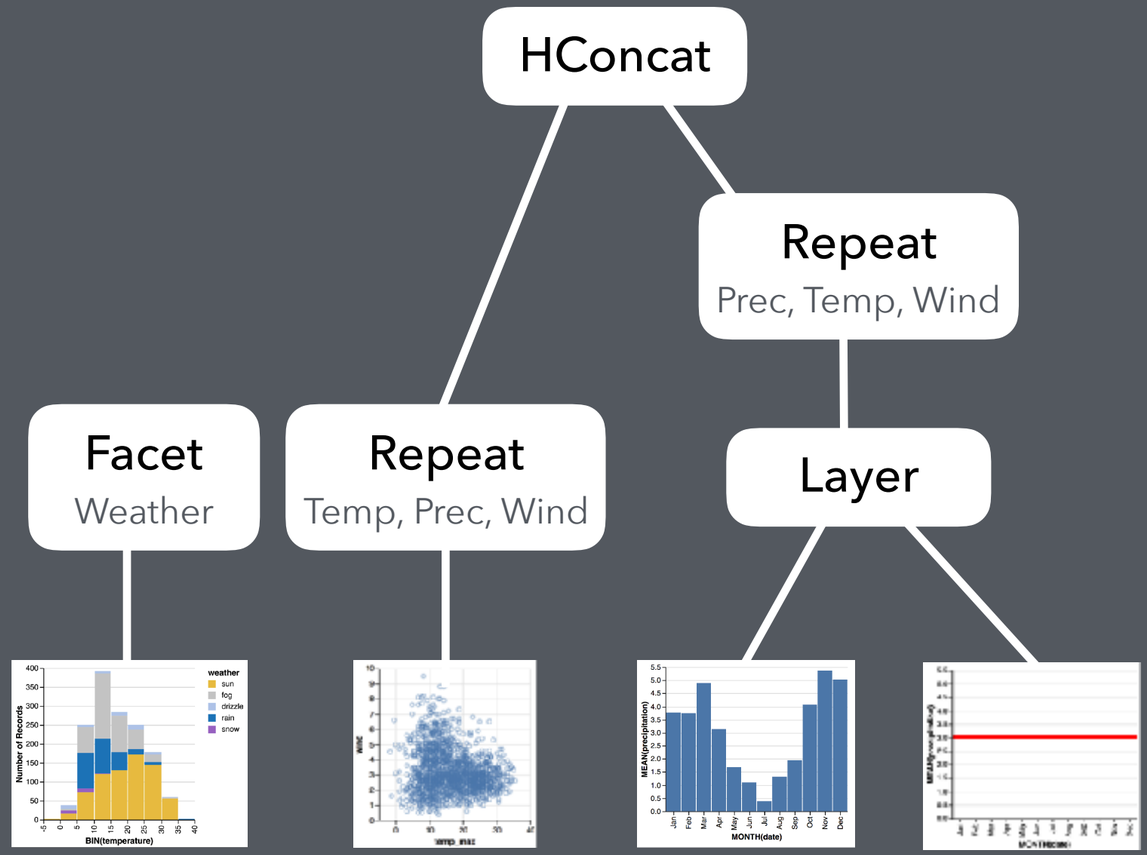


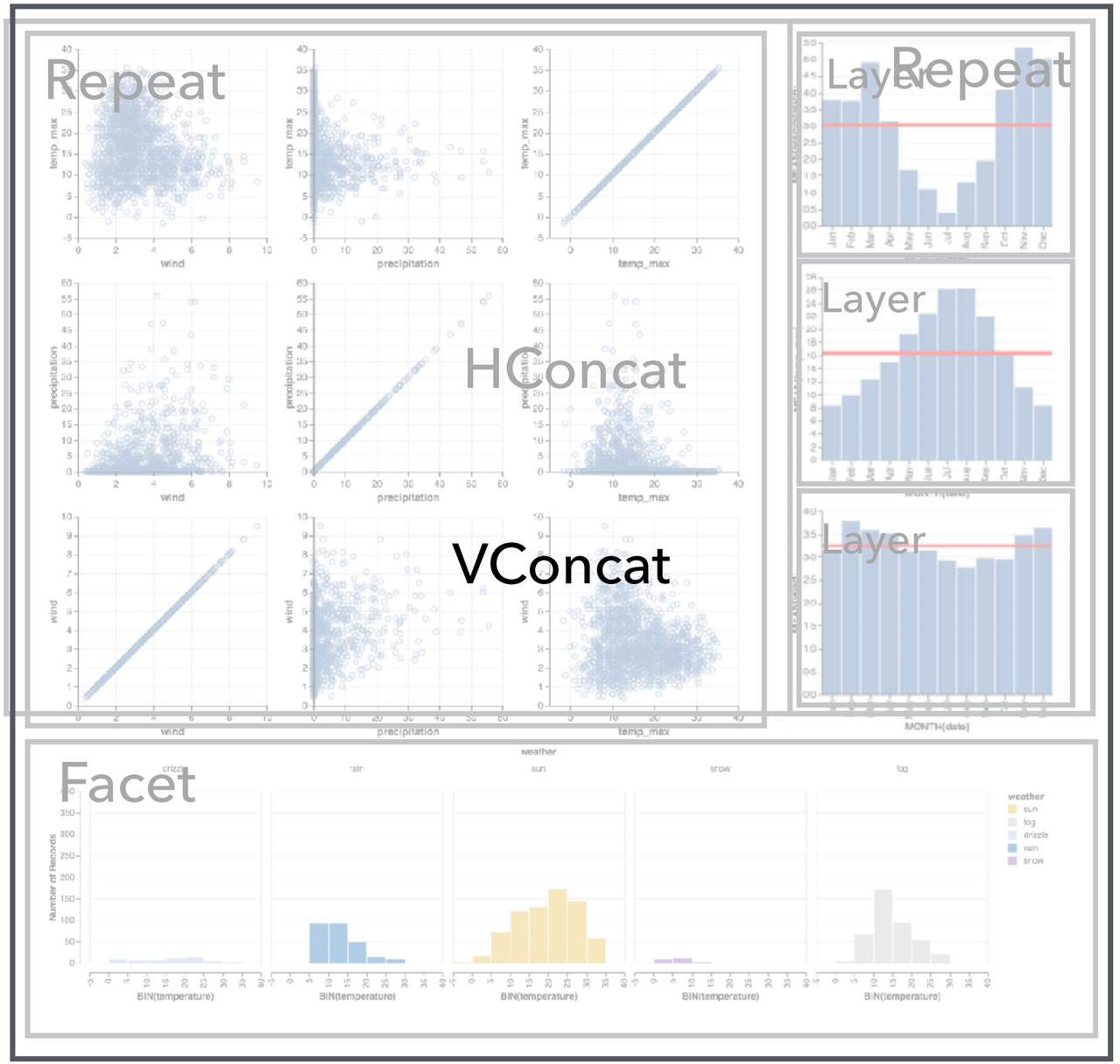




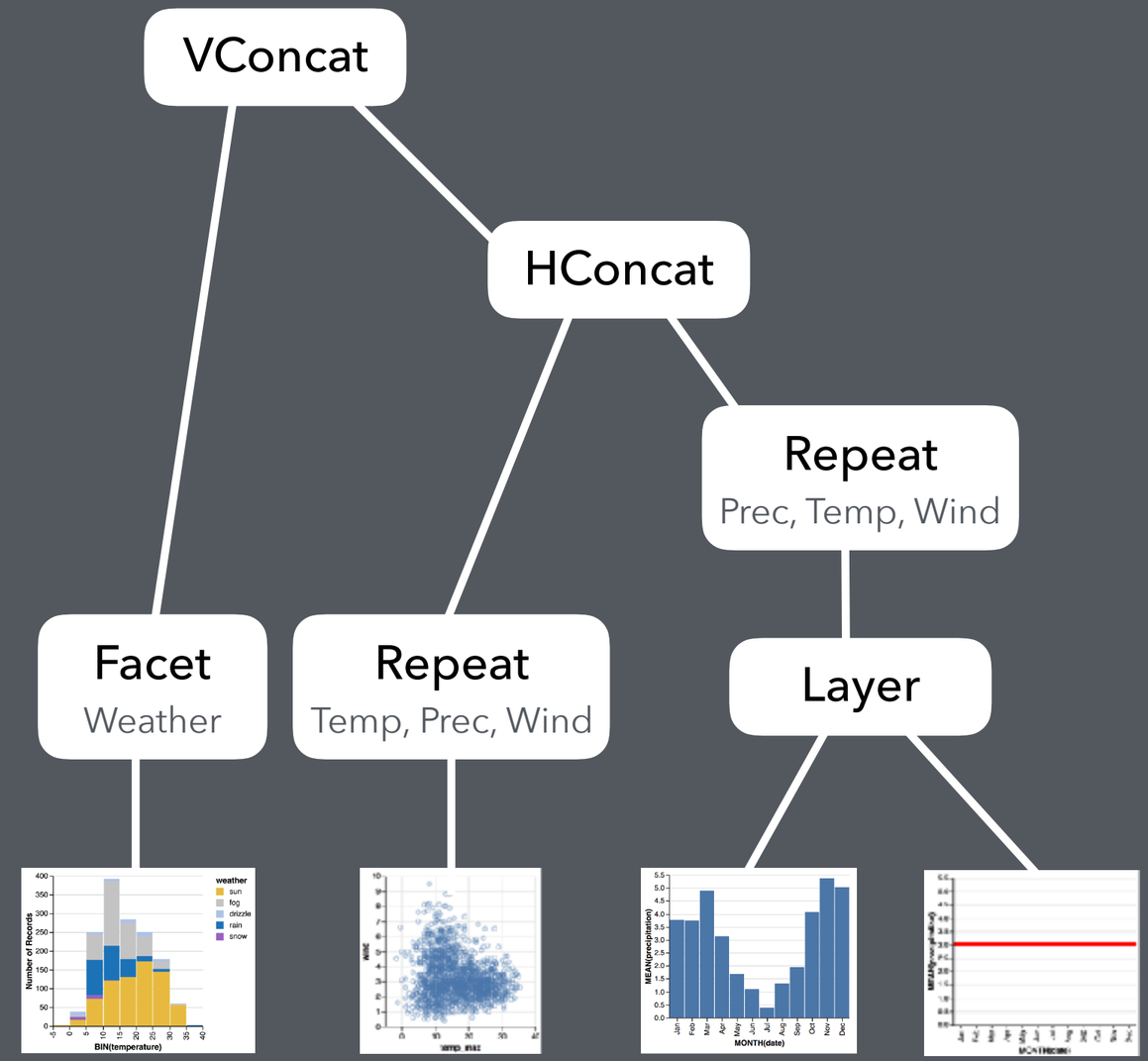
Hierarchical View Composition

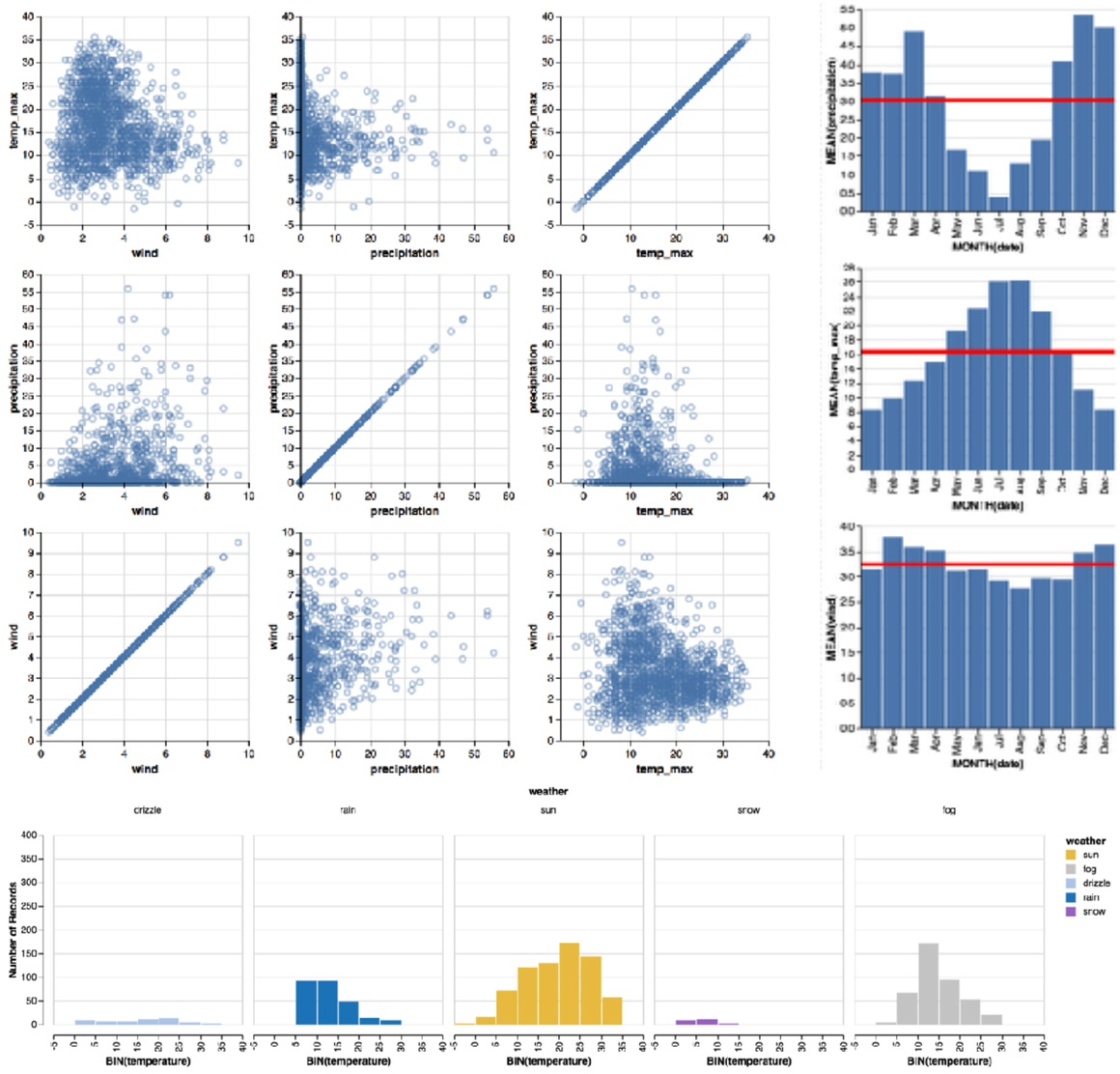




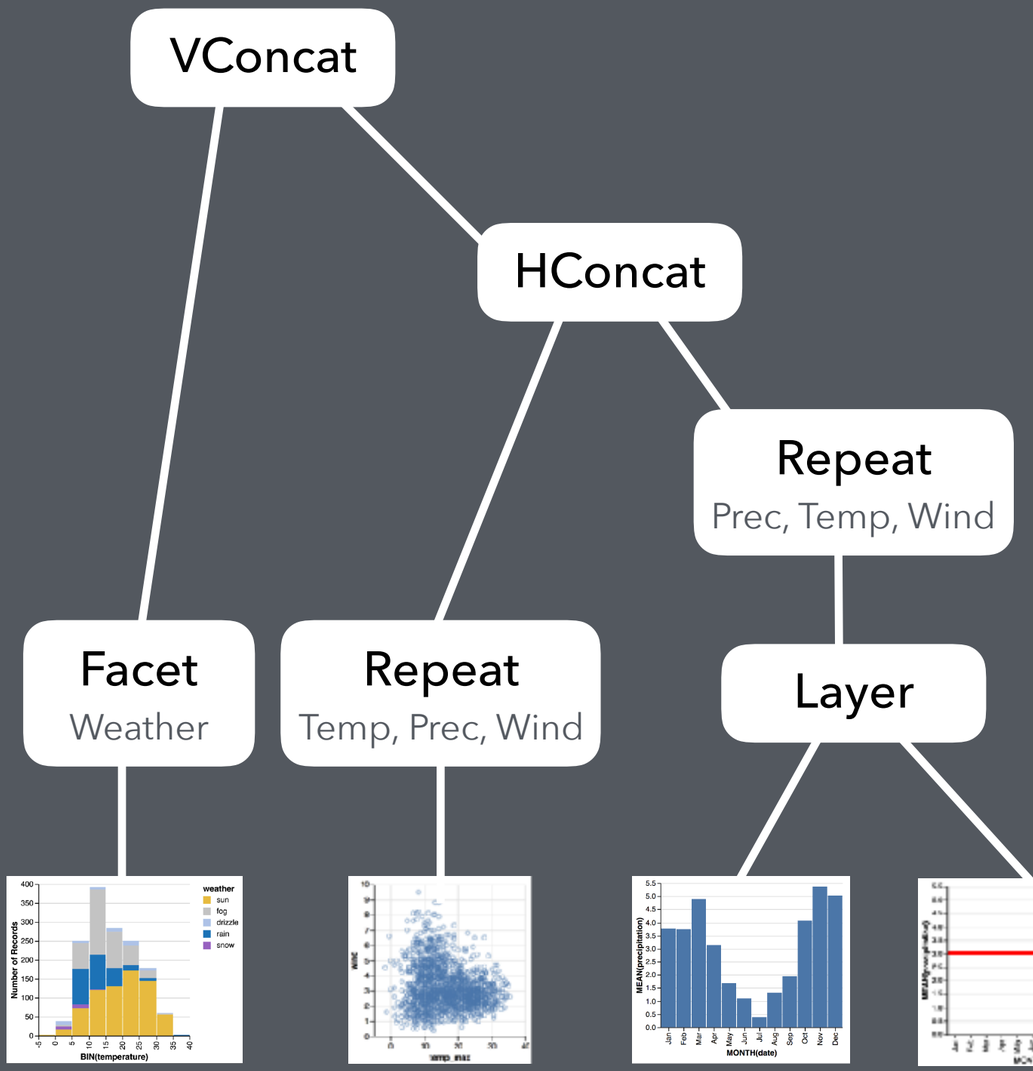


Hierarchical View Composition

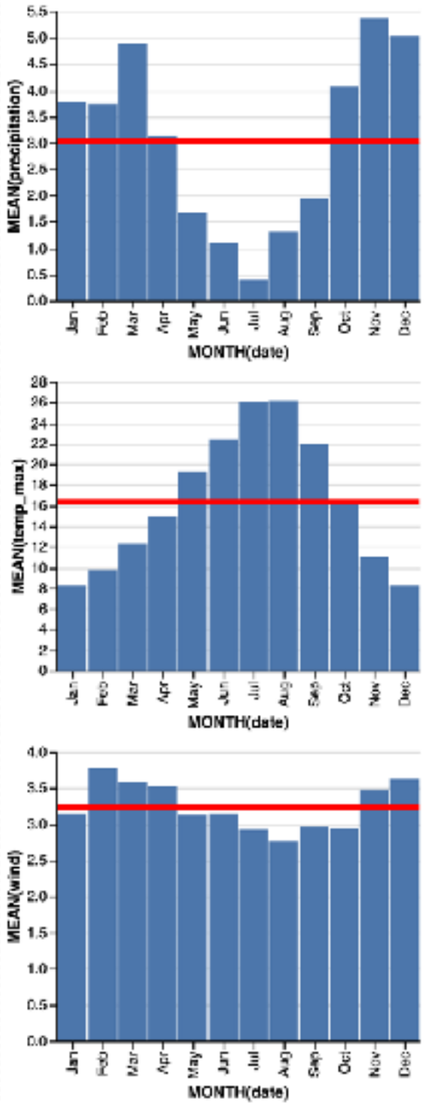




Hierarchical View Composition



Layer + Repeat



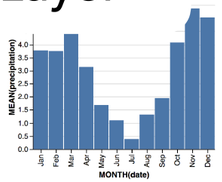

```
{
```

Repeat

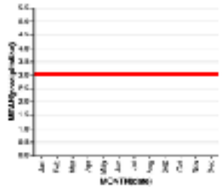
Prec, Temp, Wind `'precipitation", "temperature", "wind"]`

```
  },
  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

Using Vega-Lite

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

Selections

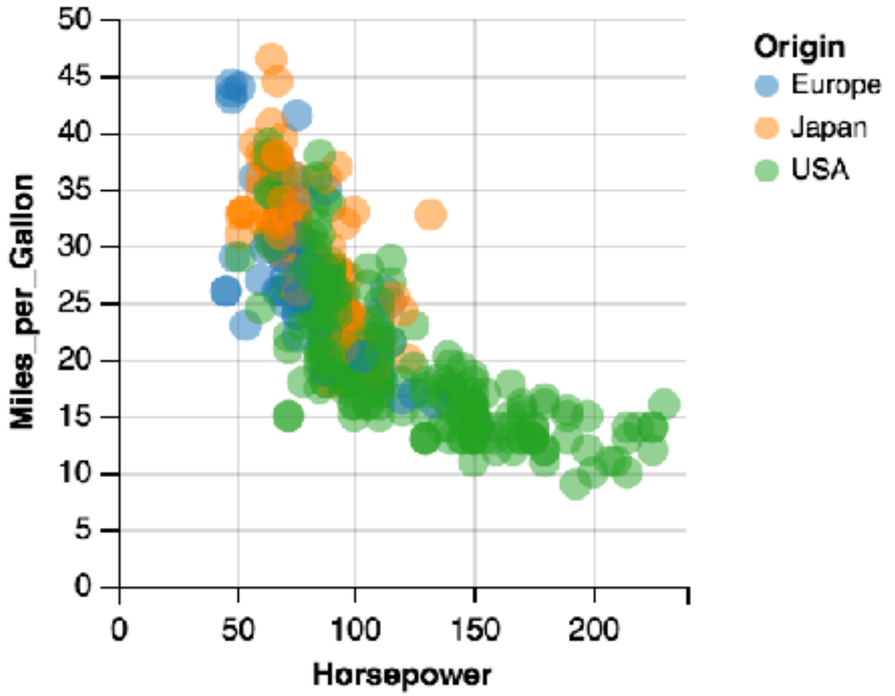
Selections

The core interactive building blocks. Define three components:

1. Event processing – how does the interaction occur?
2. Points of interest – which marks/data points were interacted with?
3. Predicate function – what is the full set of selected marks/data points?

Vega-Lite Selections

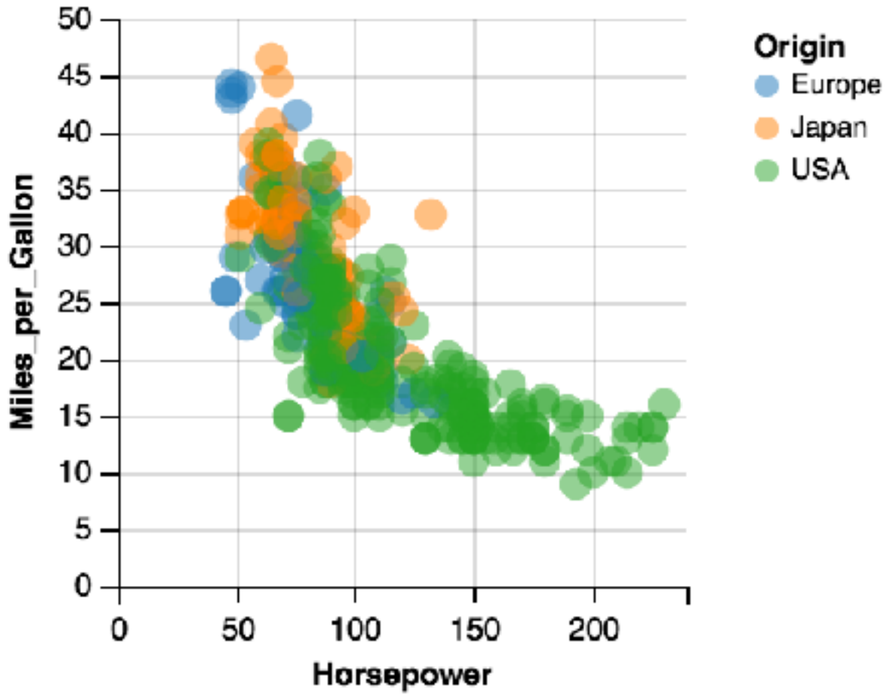
```
{
  data: {url: "data/cars.json"},
  mark: "circle",
  encoding: {
    x: {field: "Horsepower", type: "Q"},
    y: {field: "Miles_per_Gallon", type: "Q"},
    color: {field: "Origin", type: "N"}
  }
}
```



Selections define event processing, points of interest, and a predicate function.

Vega-Lite Selections: A Single Point

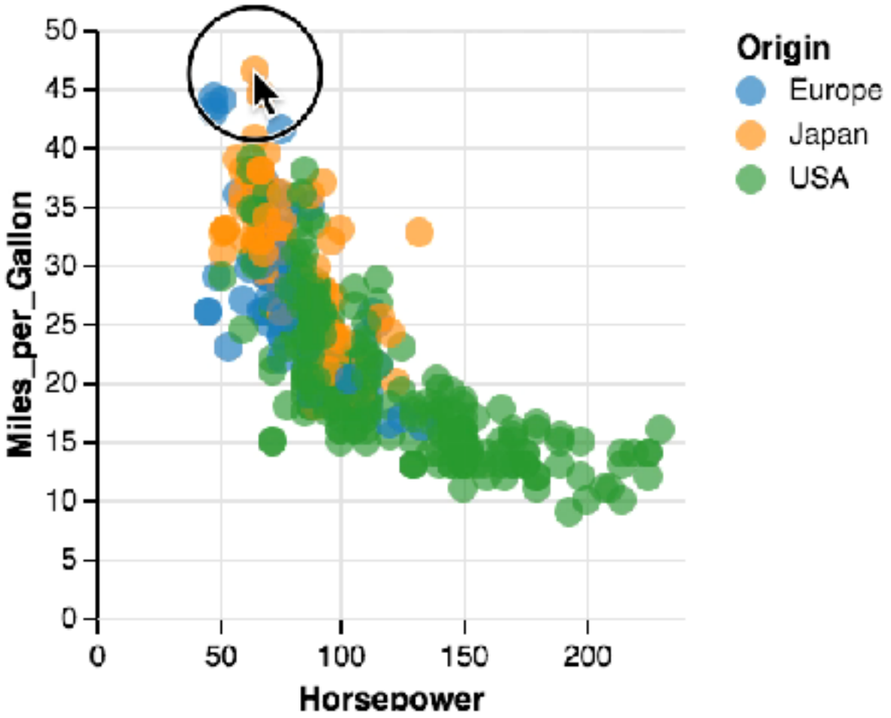
```
{
  data: {url: "data/cars.json"},
  mark: "circle",
  selection: {
    picked: {type: "single"}
  },
  encoding: {
    x: {field: "Horsepower", type: "Q"},
    y: {field: "Miles_per_Gallon", type: "Q"},
    color: {field: "Origin", type: "N"}
  }
}
```



Selections define event processing, points of interest, and a predicate function.

Vega-Lite Selections: A Single Point

```
{
  data: {url: "data/cars.json"},
  mark: "circle",
  selection: {
    picked: {type: "single"}
  },
  encoding: {
    x: {field: "Horsepower", type: "Q"},
    y: {field: "Miles_per_Gallon", type: "Q"},
    color: {
      condition: {
        selection: "!picked", value: "grey"
      },
      field: "Origin", type: "N"
    }
  }
}
```

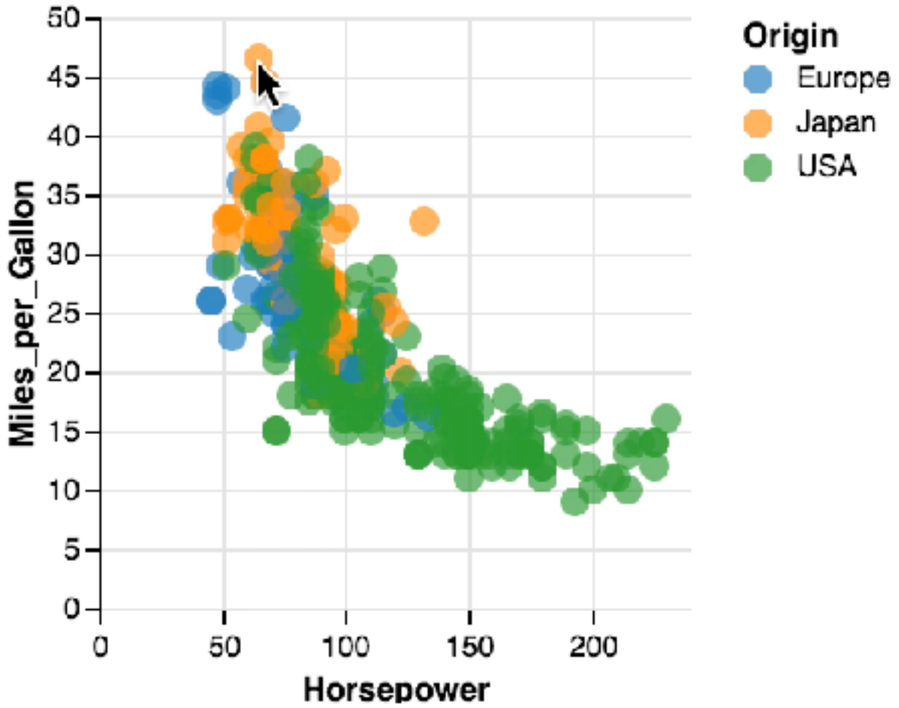


Selections define event processing, points of interest, and a predicate function.

Selection **types** provide defaults values for these three components.

Vega-Lite Selections: Multiple Points

```
{
  data: {url: "data/cars.json"},
  mark: "circle",
  selection: {
    picked: {type: "multi"}
  },
  encoding: {
    x: {field: "Horsepower", type: "Q"},
    y: {field: "Miles_per_Gallon", type: "Q"},
    color: {
      condition: {
        selection: "!picked", value: "grey"
      },
      field: "Origin", type: "N"
    }
  }
}
```

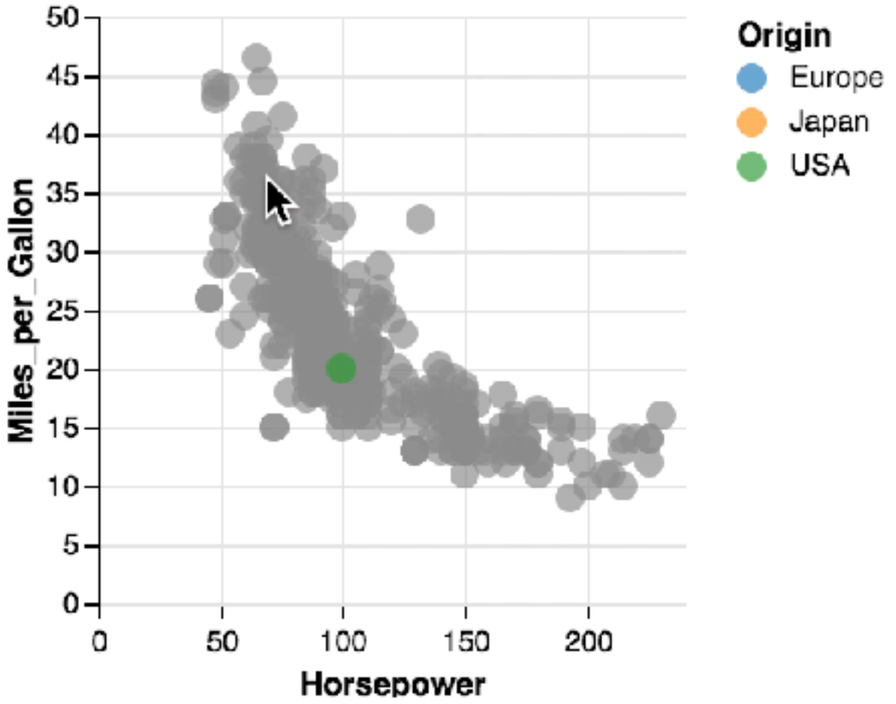


Selections define event processing, points of interest, and a predicate function.

Selection **types** provide defaults values for these three components.

Vega-Lite Selections: Multiple Points on hover

```
{
  data: {url: "data/cars.json"},
  mark: "circle",
  selection: {
    picked: {type: "multi", on: "mouseover"}
  },
  encoding: {
    x: {field: "Horsepower", type: "Q"},
    y: {field: "Miles_per_Gallon", type: "Q"},
    color: {
      condition: {
        selection: "!picked", value: "grey"
      },
      field: "Origin", type: "N"
    }
  }
}
```



Selections define event processing, points of interest, and a predicate function.

Selection **types** provide defaults values for these three components.

Selections

The core interactive building blocks. Define three components:

1. Event processing – how does the interaction occur?
2. Points of interest – which marks/data points were interacted with?
3. Predicate function – what is the full set of selected marks/data points?

Selections

The core interactive building blocks. Define three components:

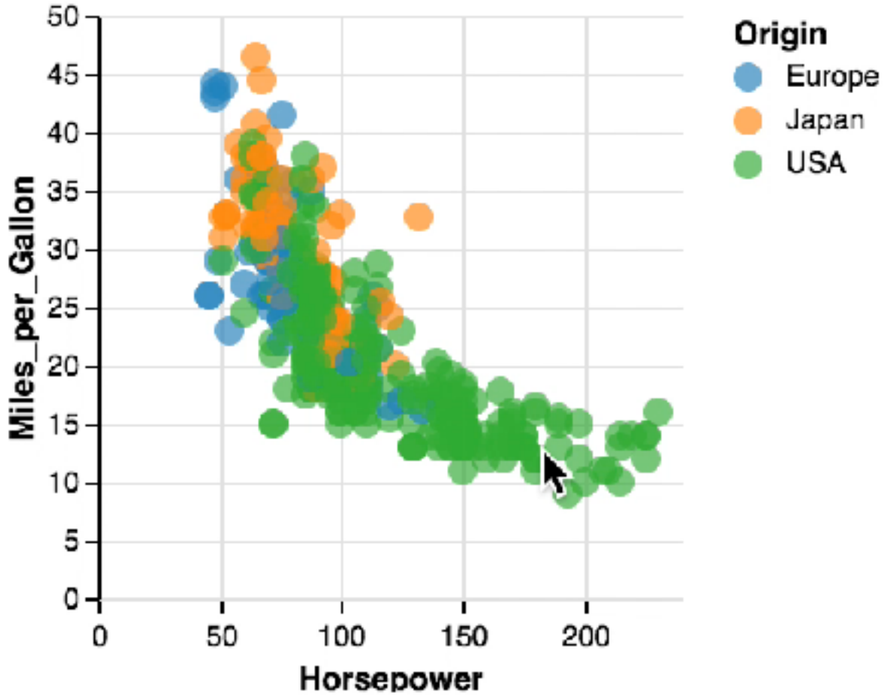
1. Event processing – how does the interaction occur?
2. Points of interest – which marks/data points were interacted with?
3. Predicate function – what is the full set of selected marks/data points?

Selection Transforms

Composable operators that modify a selection's components.

Vega-Lite Selections: A Single Cylinder

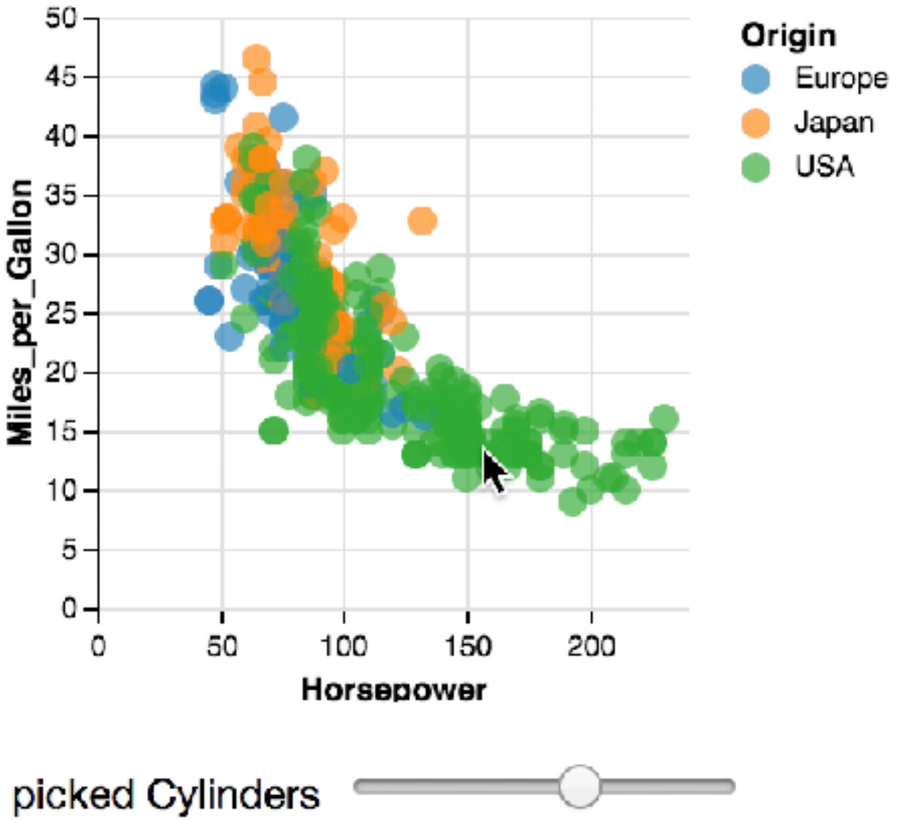
```
{
  data: {url: "data/cars.json"},
  mark: "circle",
  selection: {
    picked: {
      type: "single", fields: ["Cylinders"]
    }
  },
  encoding: {
    x: {field: "Horsepower", type: "Q"},
    y: {field: "Miles_per_Gallon", type: "Q"},
    color: {
      condition: {
        selection: "!picked", value: "grey"
      },
      field: "Origin", type: "N"
    }
  }
}
```



The **project transform** rewrites the predicate to match on **`fields`** or `encodings`.

Vega-Lite Selections: A Single Cylinder & Year

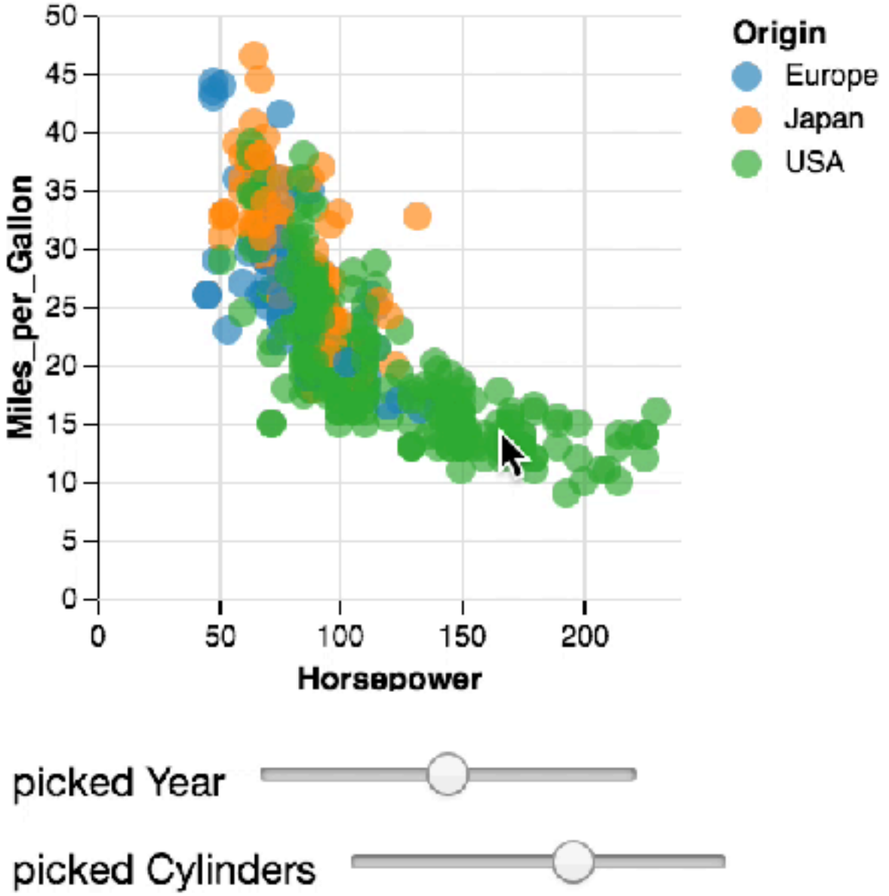
```
{
  data: {url: "data/cars.json"},
  mark: "circle",
  selection: {
    picked: {
      type: "single", fields: ["Cylinders"],
      bind: {input: "range", min: 3, ...}
    }
  },
  encoding: {
    x: {field: "Horsepower", type: "Q"},
    y: {field: "Miles_per_Gallon", type: "Q"},
    color: {
      condition: {
        selection: "!picked", value: "grey"
      },
      field: "Origin", type: "N"
    }
  }
```



The **bind transform** drives selections via **query widgets** and scale functions.

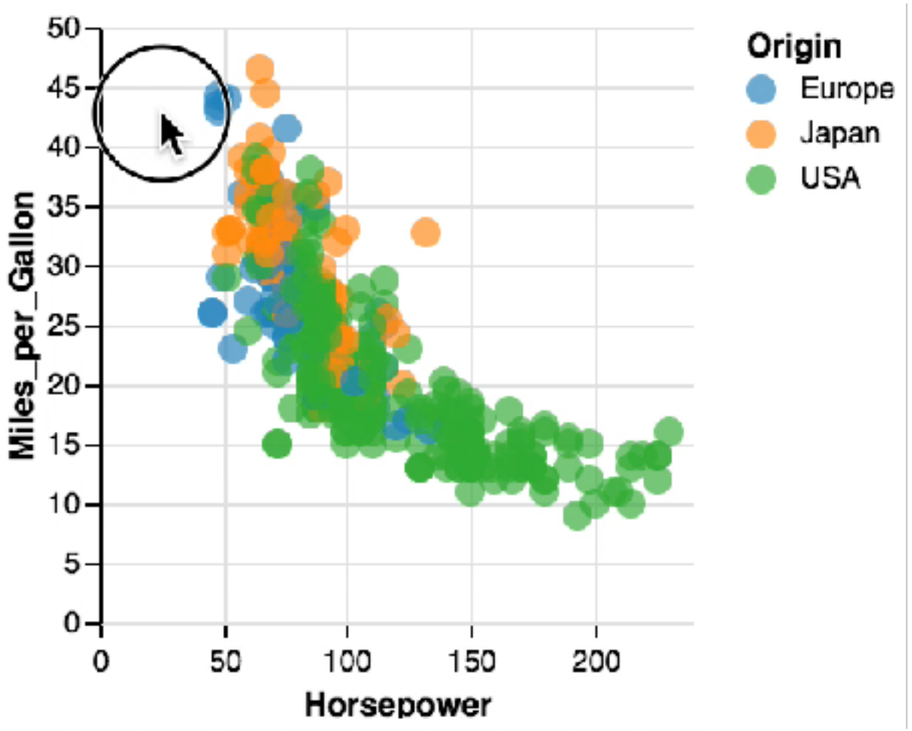
Vega-Lite Selections: A Single Cylinder & Year

```
{
  data: {url: "data/cars.json"},
  mark: "circle",
  selection: {
    picked: {
      type: "single",
      fields: ["Cylinders", "Year"],
      bind: {
        Cylinders: {input: "range", min: 3},
        Year: {input: "range", min: 1967}
      }
    }
  },
  encoding: {
    x: {field: "Horsepower", type: "Q"},
    y: {field: "Miles_per_Gallon", type: "Q"},
    color: {
      condition: {
        selection: "!picked", value: "grey"
      },
      field: "Origin", type: "N"
    }
```



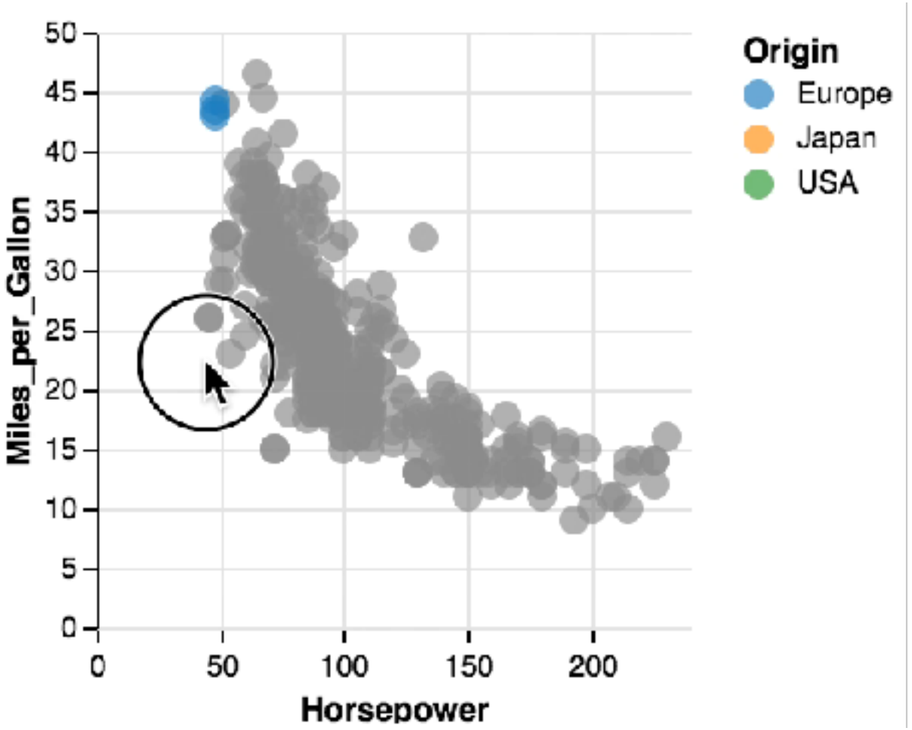
Vega-Lite Selections: Continuous Region

```
{
  data: {url: "data/cars.json"},
  mark: "circle",
  selection: {
    grid: {type: "interval"}
  },
  encoding: {
    x: {field: "Horsepower", type: "Q"},
    y: {field: "Miles_per_Gallon", type: "Q"},
    color: {
      condition: {
        selection: "!grid", value: "grey"
      },
      field: "Origin", type: "N"
    }
  }
}
```



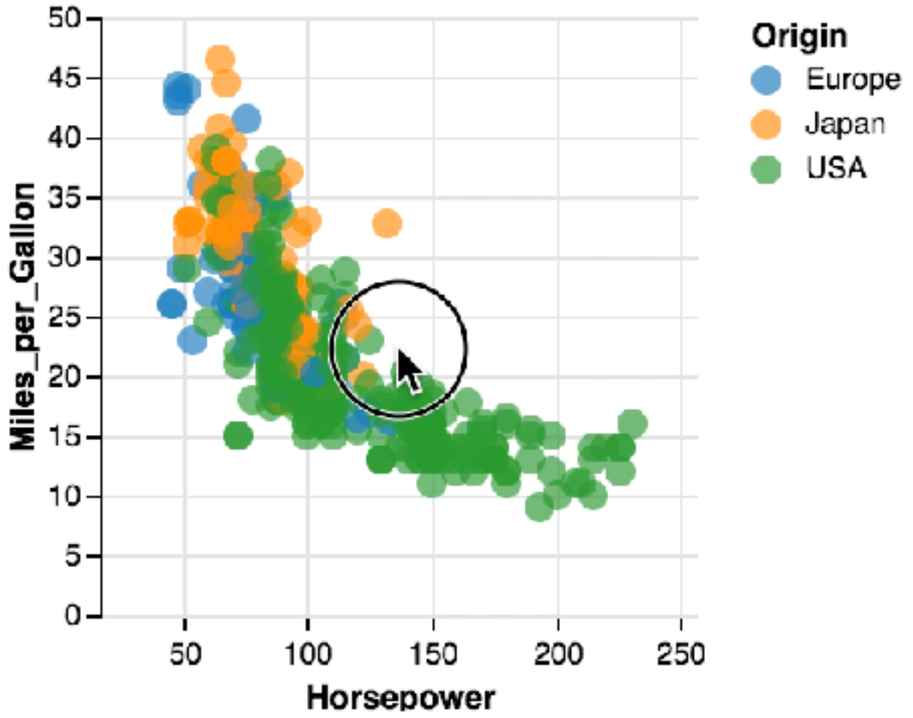
Vega-Lite Selections: Continuous Region

```
{
  data: {url: "data/cars.json"},
  mark: "circle",
  selection: {
    grid: {
      type: "interval",
      encodings: ["x"]

    }
  },
  encoding: {
    x: {field: "Horsepower", type: "Q"},
    y: {field: "Miles_per_Gallon", type: "Q"},
    color: {
      condition: {
        selection: "!grid", value: "grey"
      },
      field: "Origin", type: "N"
    }
  }
}
```



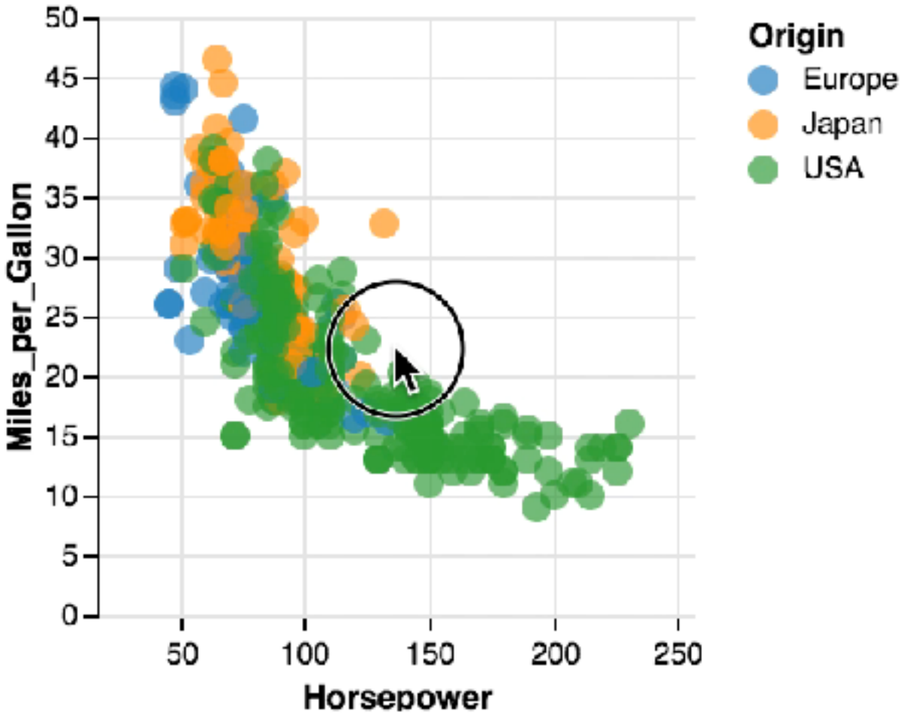
Vega-Lite Selections: Continuous Region

```
{
  data: {url: "data/cars.json"},
  mark: "circle",
  selection: {
    grid: {
      type: "interval",
      encodings: ["x"],
      bind: "scales"
    }
  },
  encoding: {
    x: {field: "Horsepower", type: "Q"},
    y: {field: "Miles_per_Gallon", type: "Q"},
    color: {
      condition: {
        selection: "!grid", value: "grey"
      },
      field: "Origin", type: "N"
    }
  }
}
```



Vega-Lite Selections: Continuous Region

```
{
  data: {url: "data/cars.json"},
  mark: "circle",
  selection: {
    grid: {
      type: "interval",
      encodings: ["x"],
      bind: "scales"
    }
  },
  encoding: {
    x: {
      field: "Horsepower", type: "Q",
      scale: {domain: {selection: "grid"}}
    },
    y: {field: "Miles_per_Gallon", type: "Q"},
```



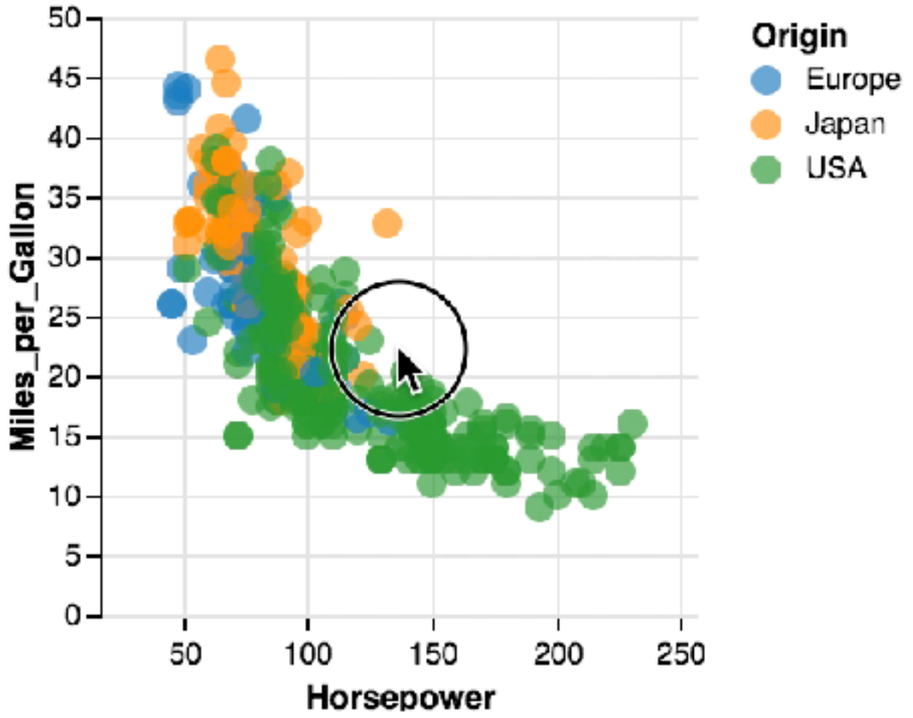
Two-way binding between a selection and scales:

Selection is populated with scale domains.

Selection now drives scale domains.

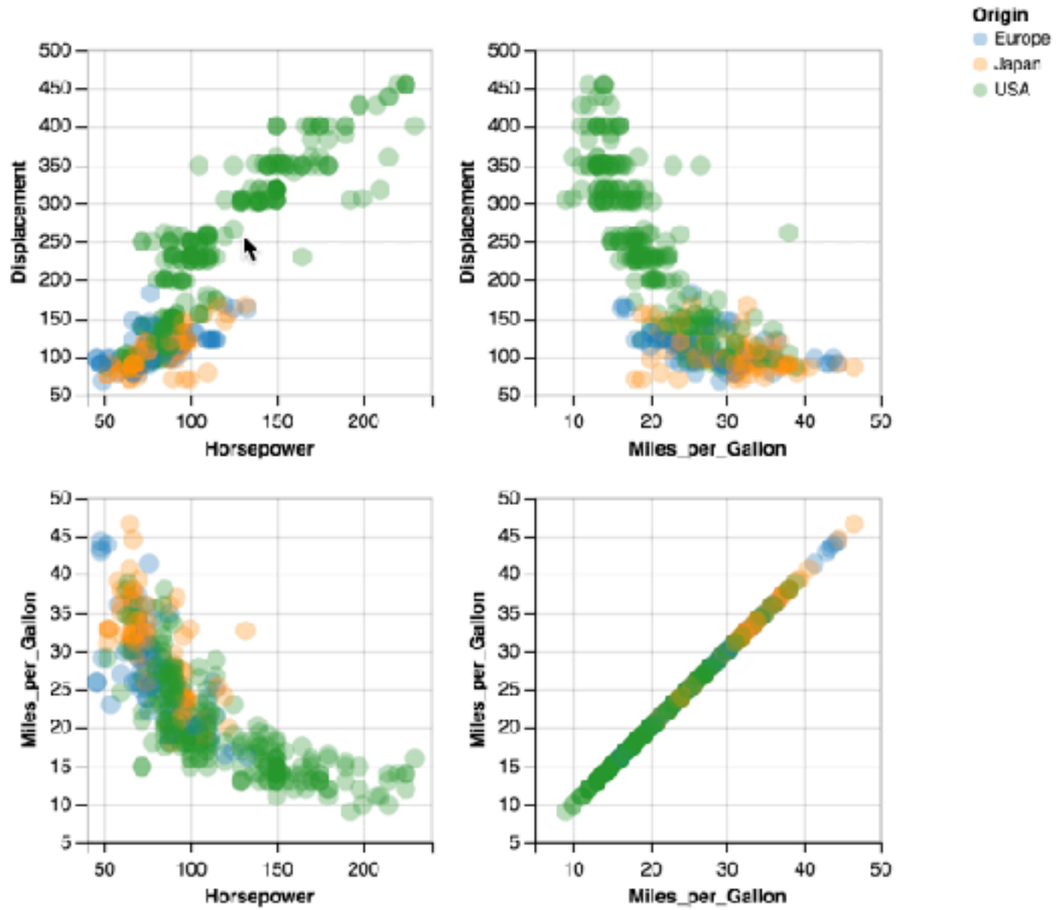
Vega-Lite Selections: Continuous Region

```
{
  data: {url: "data/cars.json"},
  mark: "circle",
  selection: {
    grid: {
      type: "interval",
      encodings: ["x"],
      bind: "scales"
    }
  },
  encoding: {
    x: {field: "Horsepower", type: "Q"},
    y: {field: "Miles_per_Gallon", type: "Q"},
    color: {
      condition: {
        selection: "!grid", value: "grey"
      },
      field: "Origin", type: "N"
    }
  }
}
```



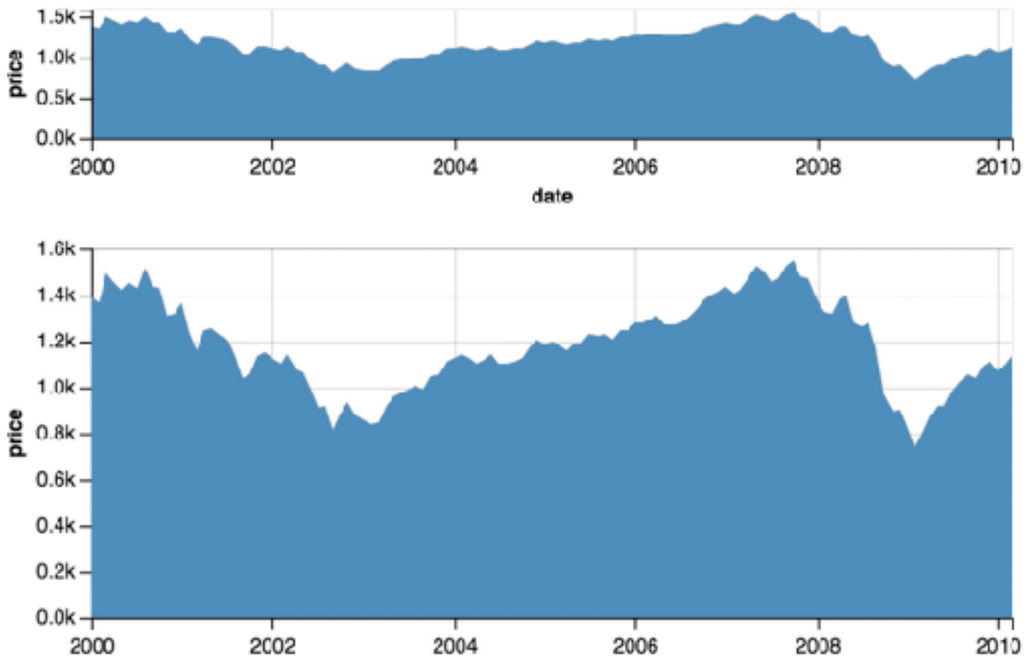
Vega-Lite Brushing & Linking

```
{
  data: {url: "data/cars.json"},
  repeat: {
    row: ["Displacement", "Miles_per_Gallon"],
    column: ["Horsepower", "Miles_per_Gallon"]
  },
  spec: {
    mark: "circle",
    selection: {
      grid: {
        type: "interval",
        encodings: ["x"],
        bind: "scales"
      }
    },
    encoding: {
      x: {field: "Horsepower", type: "Q"},
      y: {field: "Miles_per_Gallon", type: "Q"},
      color: {
        condition: {
          selection: "!grid", value: "grey"
        },
        field: "Origin", type: "N"
      }
    }
```



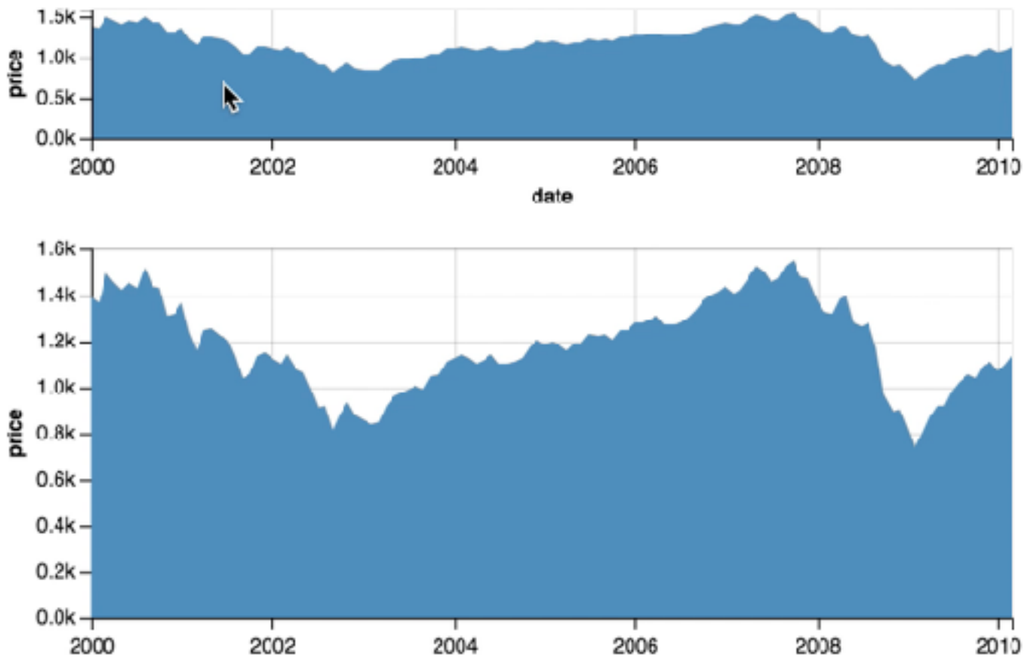
Vega-Lite Overview+Detail

```
{
  data: {url: "data/sp500.csv", ...},
  vconcat: [{
    mark: "area",
    encoding: {
      x: {field: "date", type: "T", ...},
      y: {field: "price", type: "Q", ...}
    }
  }, {
    mark: "area",
    encoding: {
      x: {field: "date", type: "T", ...},
      y: {field: "price", type: "Q", ...}
    }
  }]
}
```



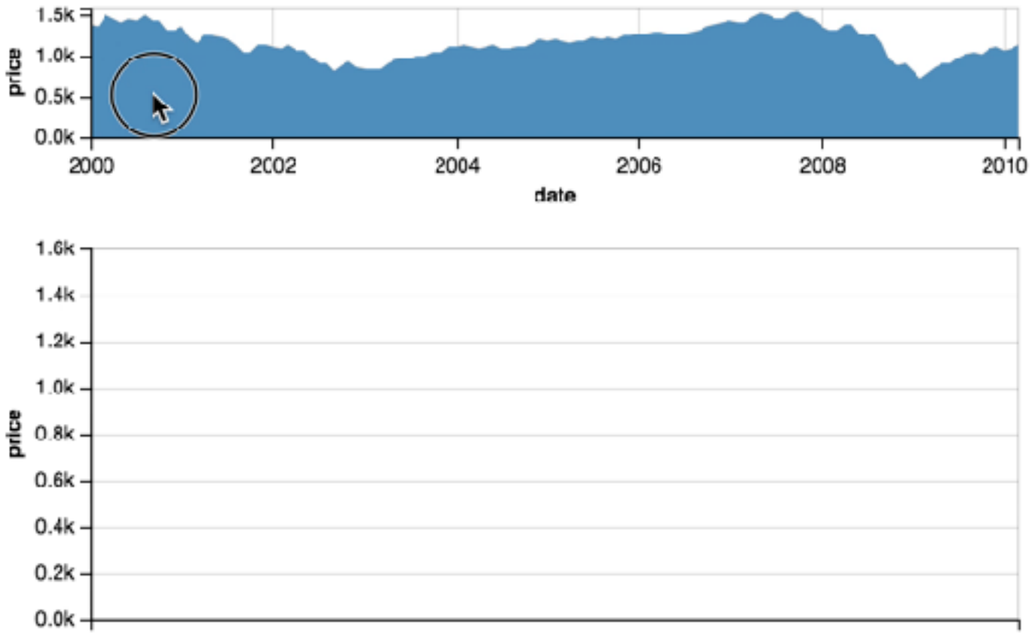
Vega-Lite Overview+Detail

```
{
  data: {url: "data/sp500.csv", ...},
  vconcat: [{
    mark: "area",
    selection: {
      region: {
        type: "interval", encodings: ["x"]
      }
    },
    ...
  }, {
    mark: "area",
    encoding: {
      x: {field: "date", type: "T", ...},
      y: {field: "price", type: "Q", ...}
    }
  }]
}
```



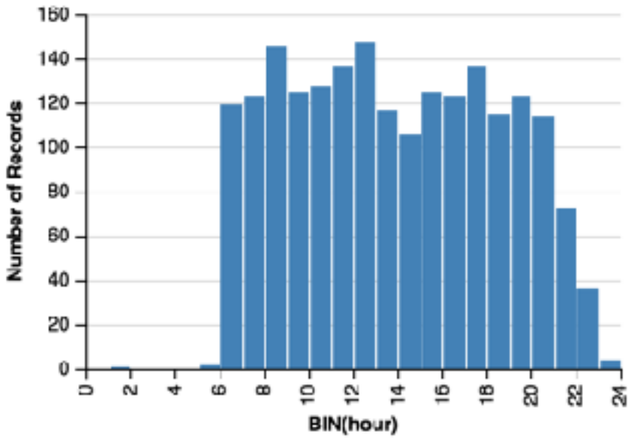
Vega-Lite Overview+Detail

```
{
  data: {url: "data/sp500.csv", ...},
  vconcat: [{
    mark: "area",
    selection: {
      region: {
        type: "interval", encodings: ["x"]
      }
    },
    ...
  }, {
    mark: "area",
    encoding: {
      x: {
        field: "date", type: "T",
        scale: {domain: {selection: "region"}}
        ...
      },
      y: {field: "price", type: "Q", ...}
    }
  }]
```



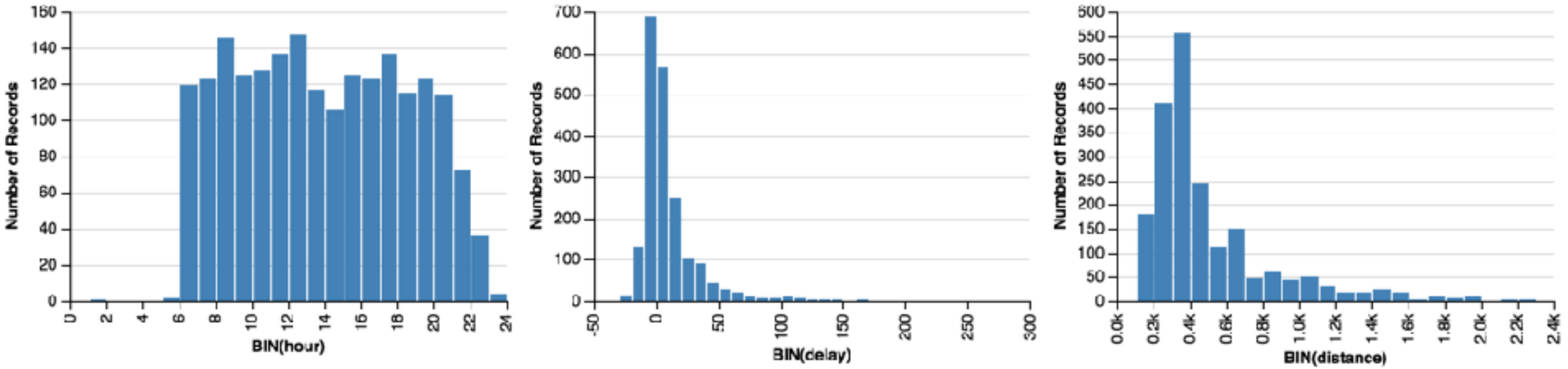
Vega-Lite Layered Cross Filtering

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



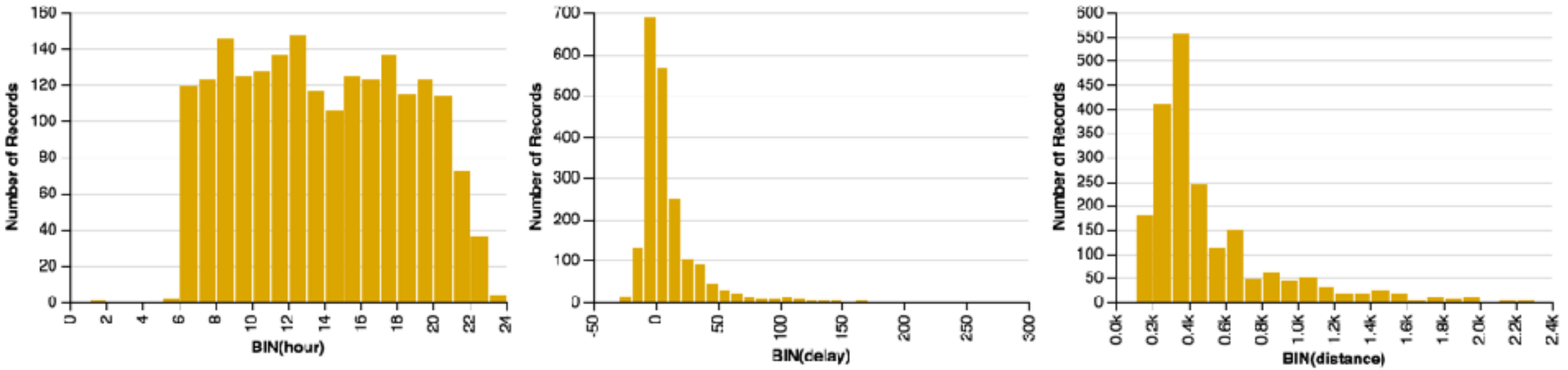
Vega-Lite Layered Cross Filtering

```
{
  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"}
    }
  }
}
```



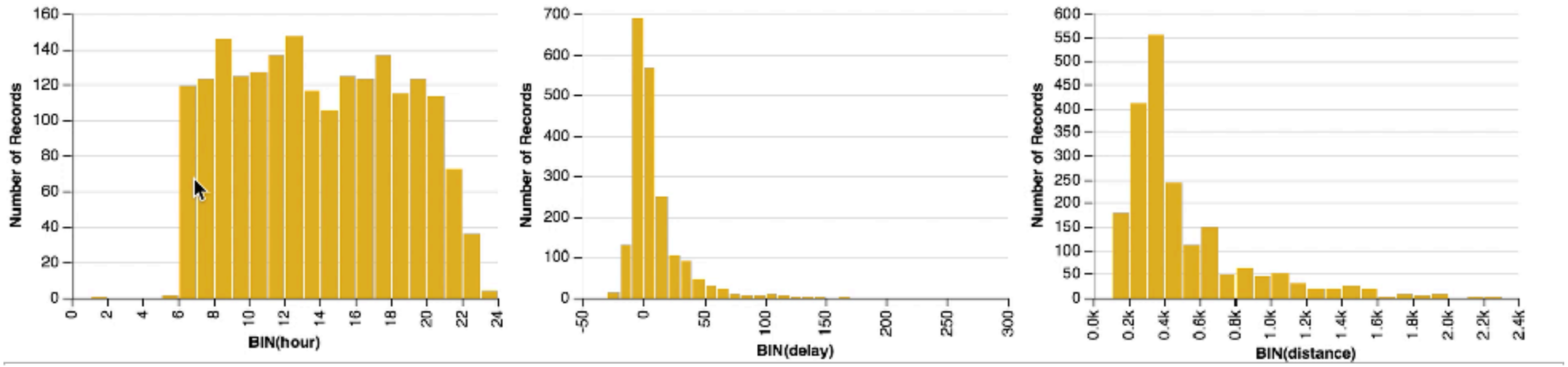
Vega-Lite Layered Cross Filtering

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



Vega-Lite Layered Cross Filtering

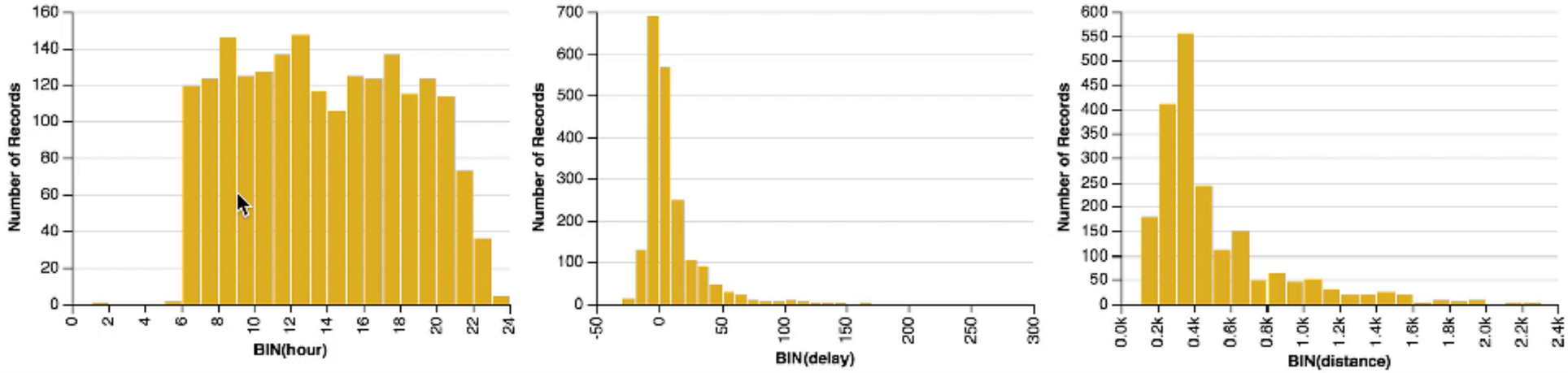
```
{
  repeat: {column: ["hour", "delay", "distance"]},
  spec: {
    layer: [{
      ...,
      selection: {
        region: {type: "interval", encodings: ["x"]}
      }
    }, {
      ...,

    }]
  }
}
```



Vega-Lite Layered Cross Filtering

```
{
  repeat: {column: ["hour", "delay", "distance"]},
  spec: {
    layer: [{
      ...,
      selection: {
        region: {type: "interval", encodings: ["x"]}
      }
    }, {
      ...,
      transform: [{filter: {selection: "region"}}]
    }]
  }
}
```

35 Lines of JSON!



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

Compile to Vega and use Vega's runtime

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

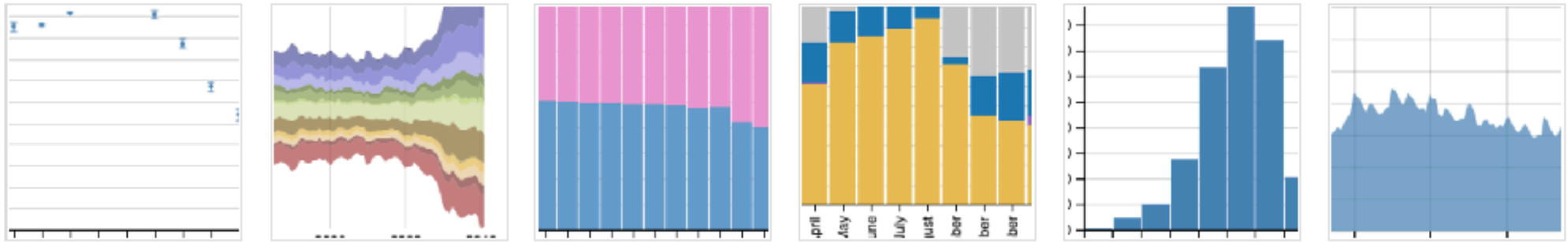
Declarative JSON Syntax

- Serve as file format
- Bindings for different languages

Altair

DECLARATIVE VISUALIZATION IN PYTHON

Declarative Visualization in Python



Altair is a declarative statistical visualization library for Python, based on **Vega-Lite**.

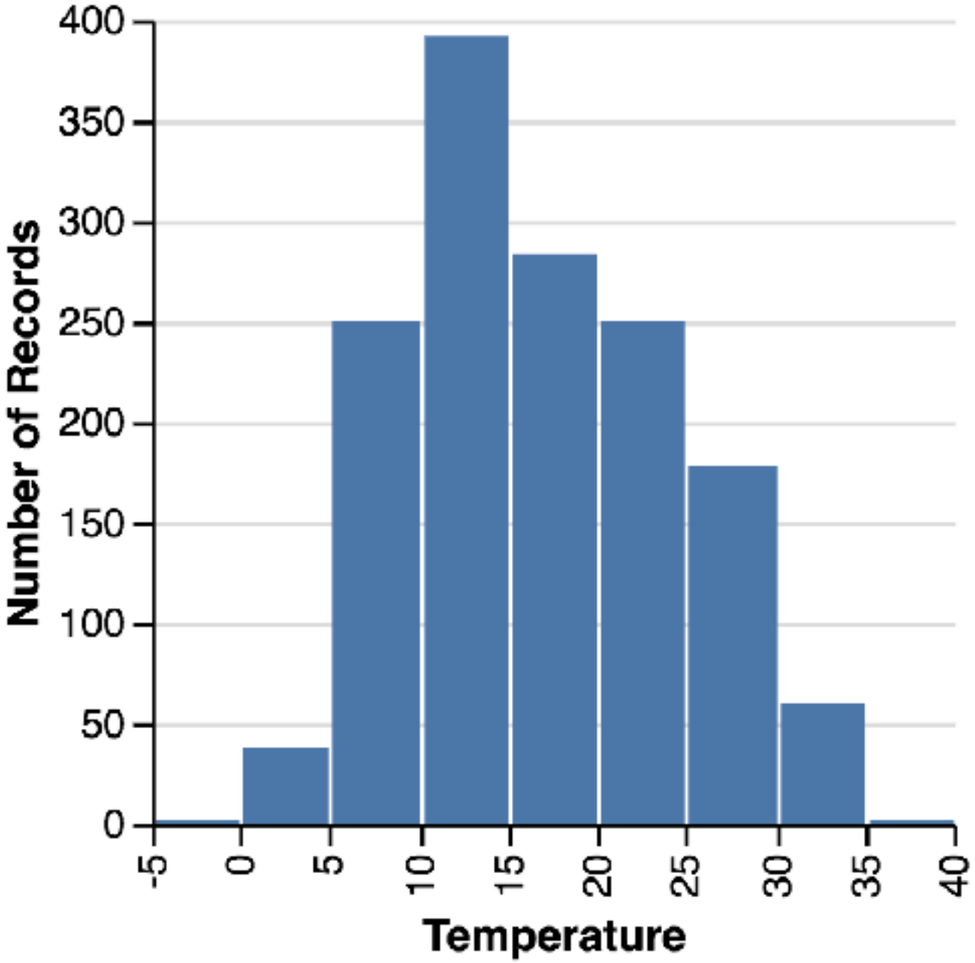
With Altair, you can spend more time understanding your data and its meaning. Altair's API is simple, friendly and consistent and built on top of the powerful **Vega-Lite** visualization grammar. This elegant simplicity produces beautiful and effective visualizations with a minimal amount of code.

Altair: Vega-Lite in Python

Led by Brian Granger and Jake VanderPlas.

Histogram in Altair

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



altair-viz.github.io

Histogram in Altair

```
{
  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

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

Chart(data_weather)
  .mark_bar()
  .encode(
     x=X(bin=True, field='Temperature'),
     y=Y(aggregate='count')
  )
```

Altair's API is automatically generated from the Vega-Lite JSON schema.

altair-viz.github.io

“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

datavoyager

Bookmarks (0) Undo Redo

Data

Cars Change

Fields

- Cylinders
- Name
- Origin
- Year
- Acceleration
- Displacement
- Horsepower
- Miles per Gallon
- Weight in lbs
- COUNT

Wildcards

- Categorical Fields
- Temporal Fields
- Quantitative Fields

Debug · Report an issue

Encoding

Clear

x drop a field here
y drop a field here
column drop a field here
row drop a field here

Marks

auto

size drop a field here
color drop a field here
shape drop a field here
detail drop a field here
text drop a field here

any drop a field here

Filter

Filter invalid numbers

Specified View

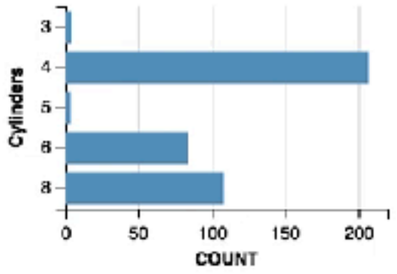
No specified visualization yet. Start exploring by dragging a field to encoding pane on the left or examining univariate summaries below.

Related Views

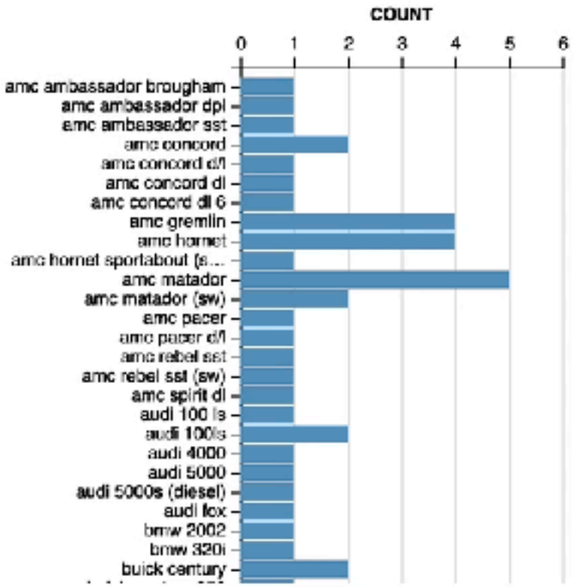
Univariate Summaries Hide

Univariate Summaries

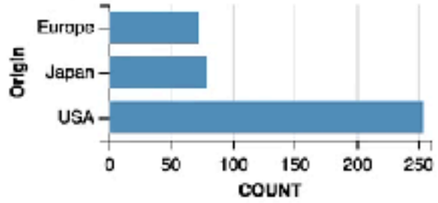
Cylinders # COUNT



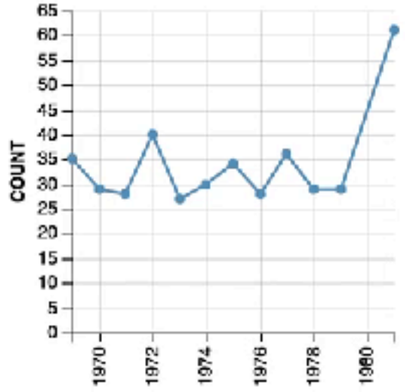
Name # COUNT



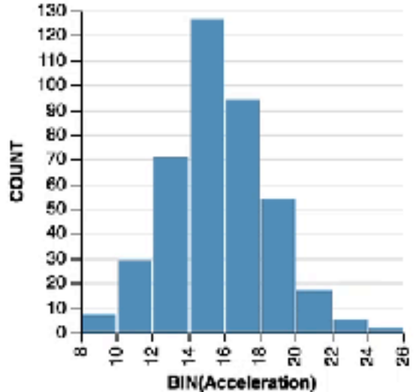
Origin # COUNT

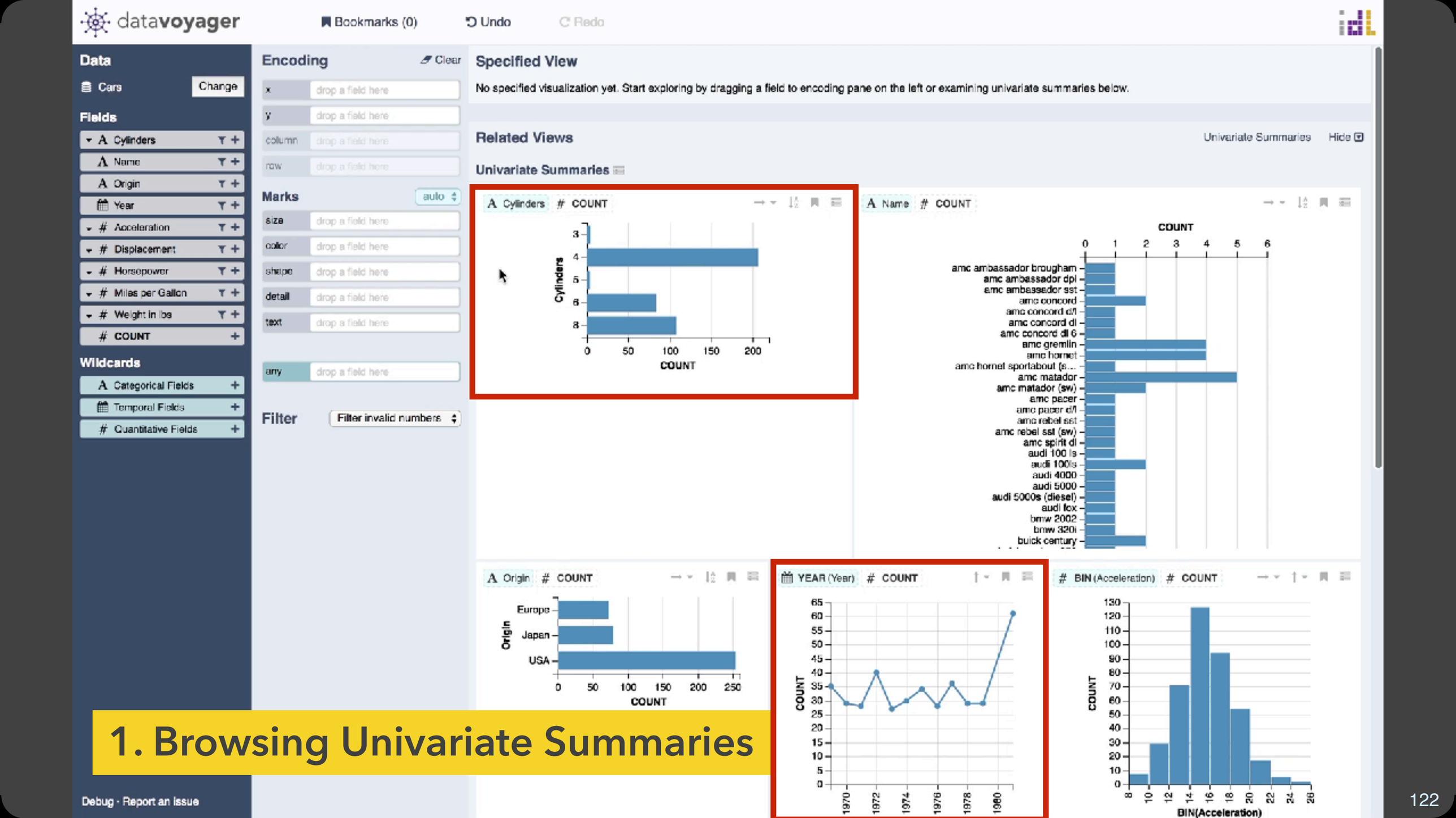


YEAR (Year) # COUNT

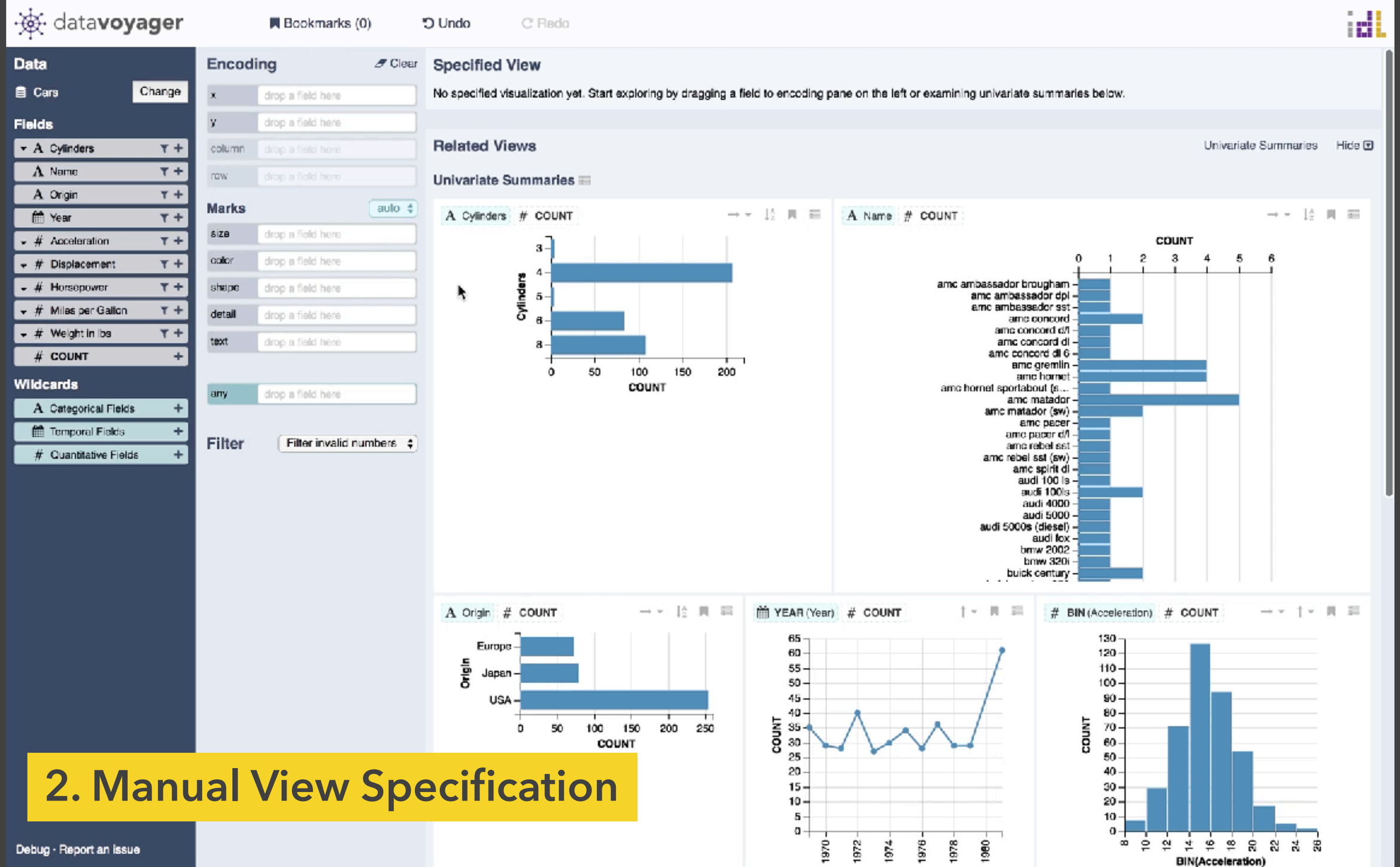


BIN (Acceleration) # COUNT



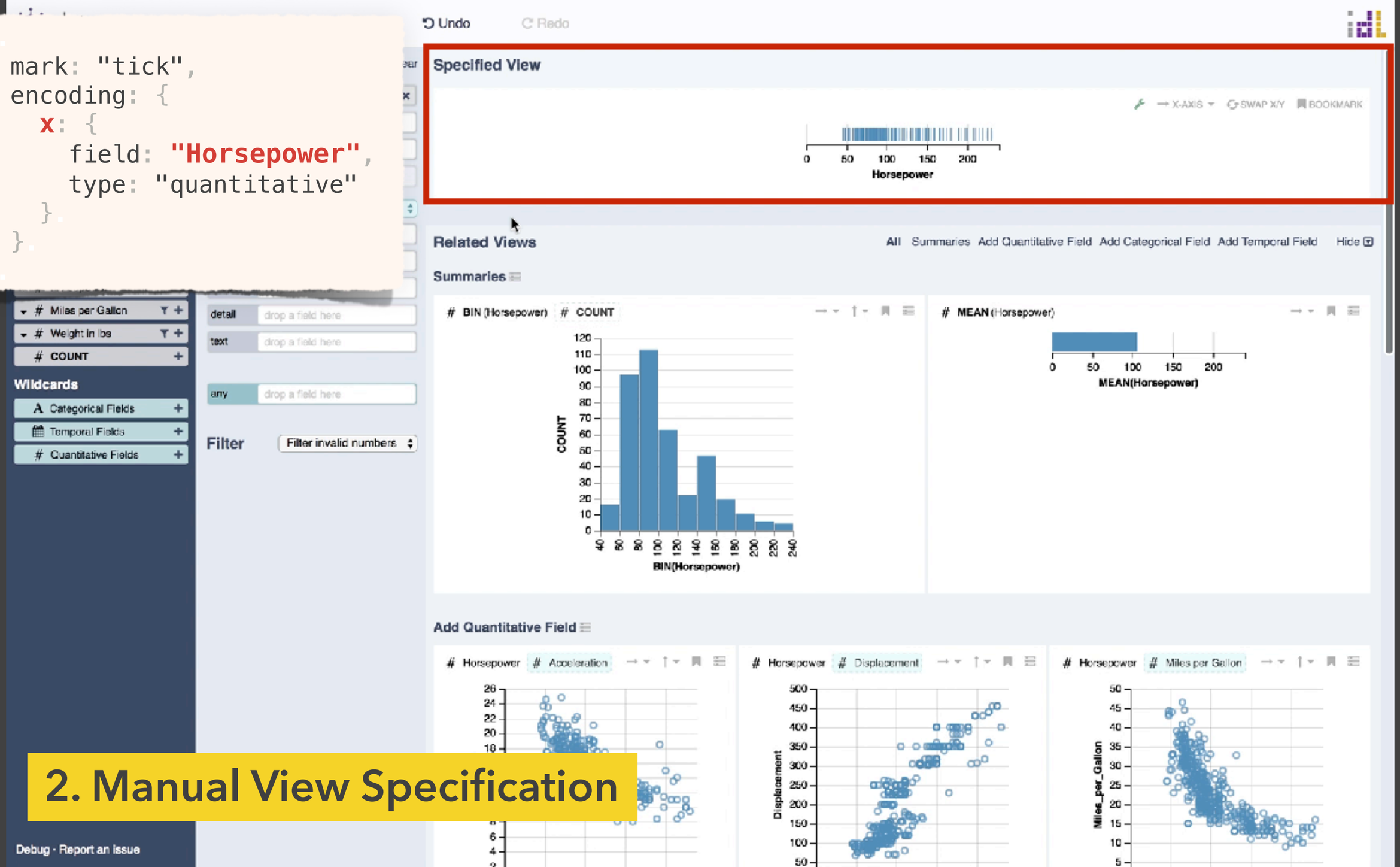


1. Browsing Univariate Summaries

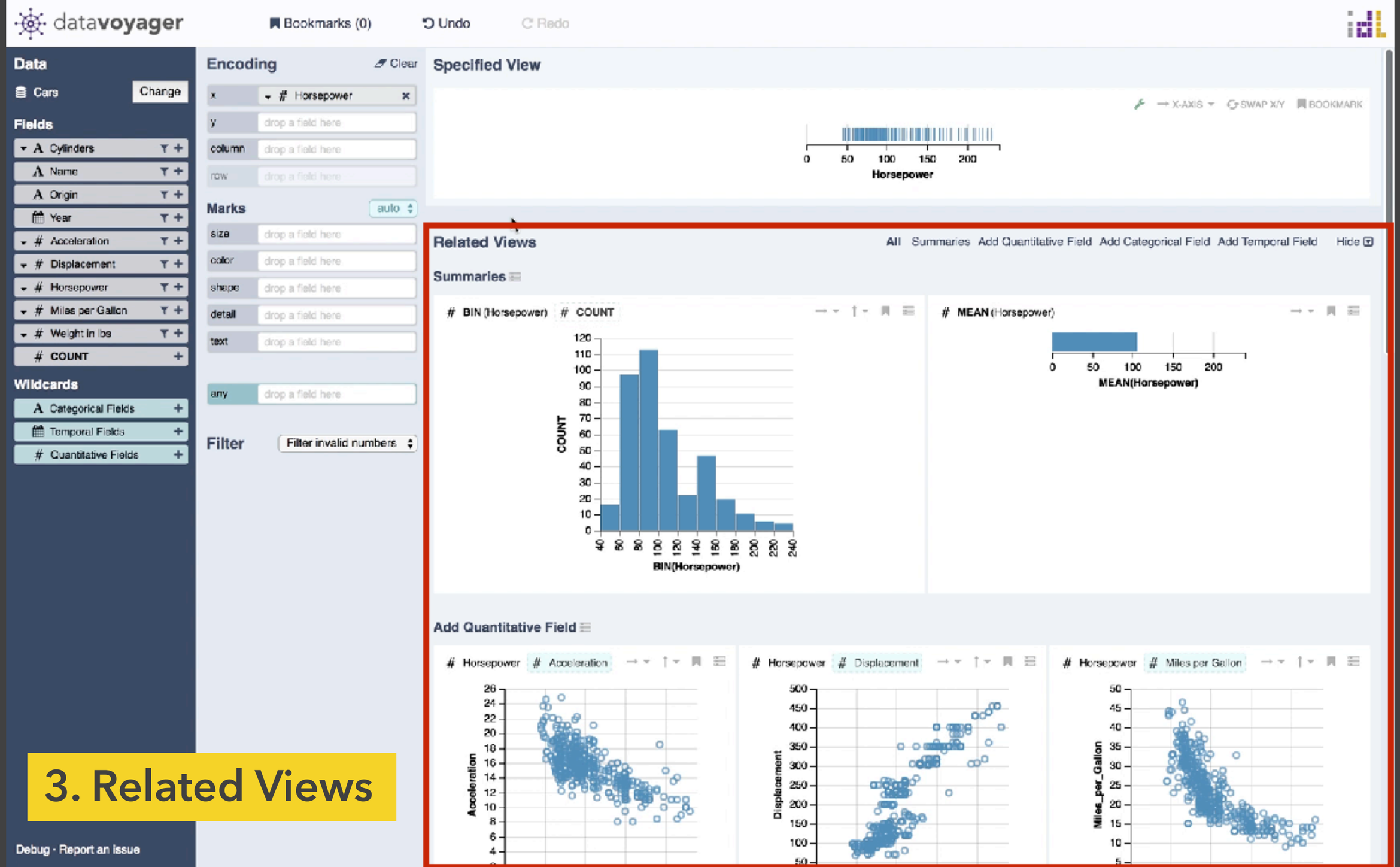


2. Manual View Specification

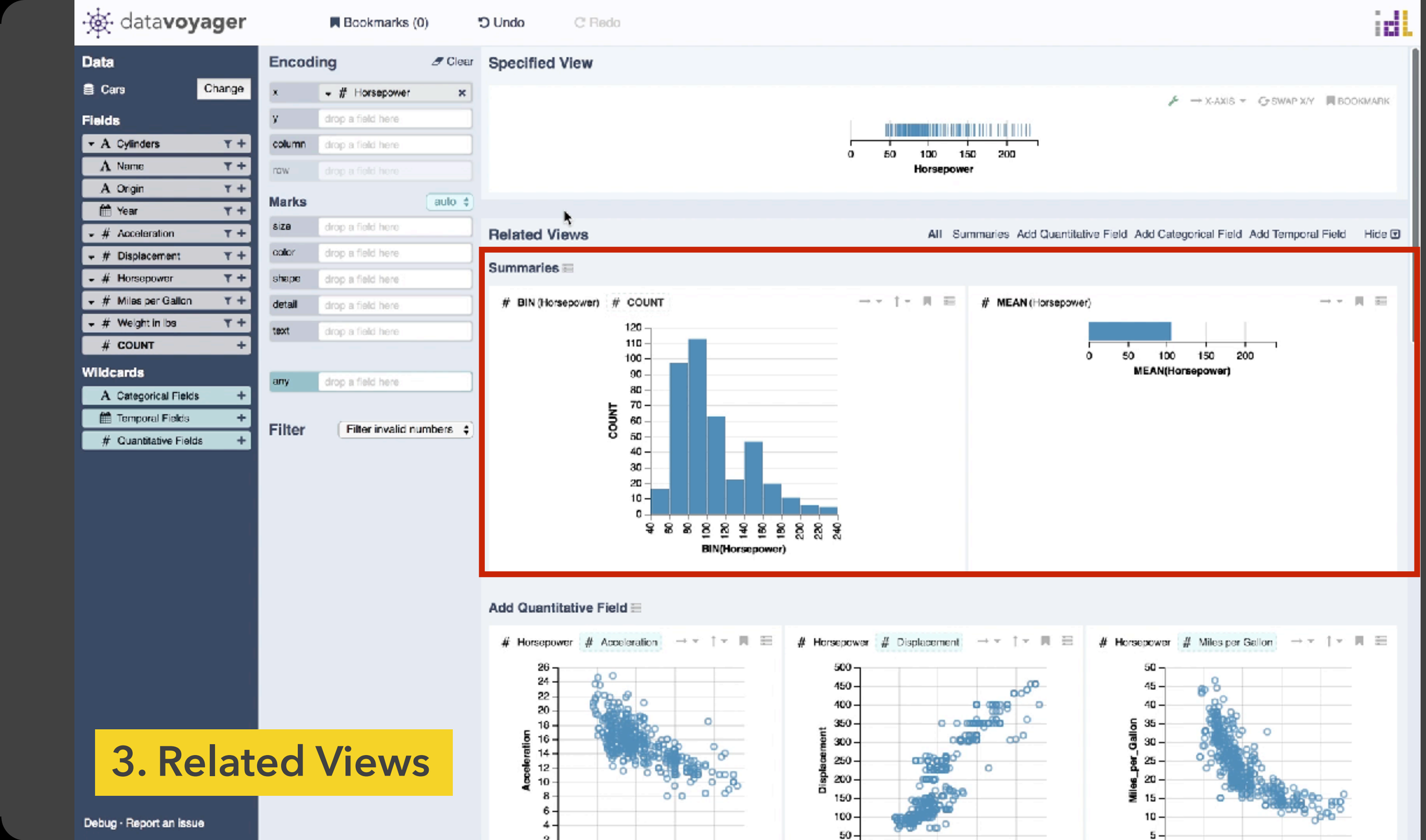
```
{
  mark: "tick",
  encoding: {
    x: {
      field: "Horsepower",
      type: "quantitative"
    }
  }
}
```



2. Manual View Specification

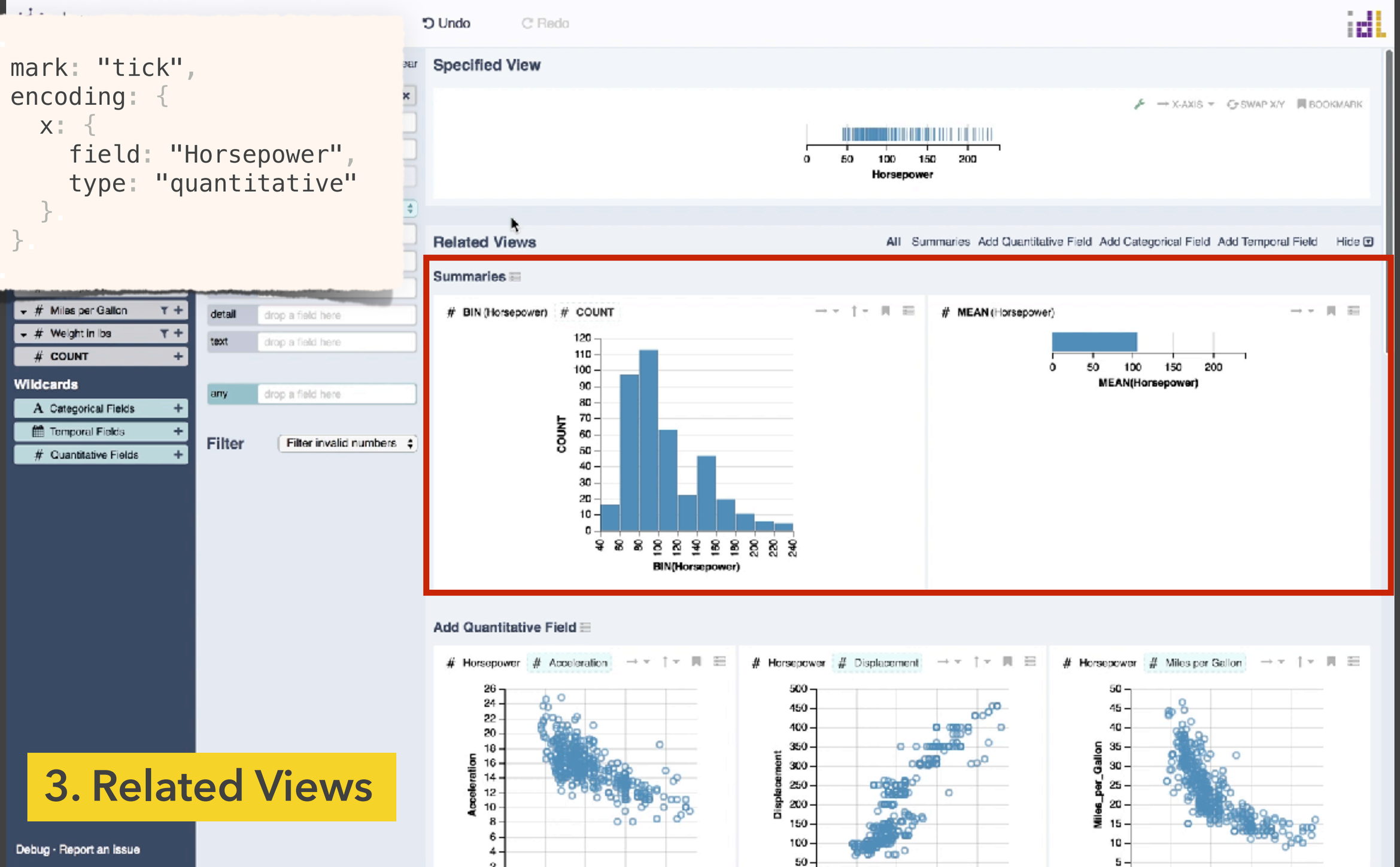


3. Related Views

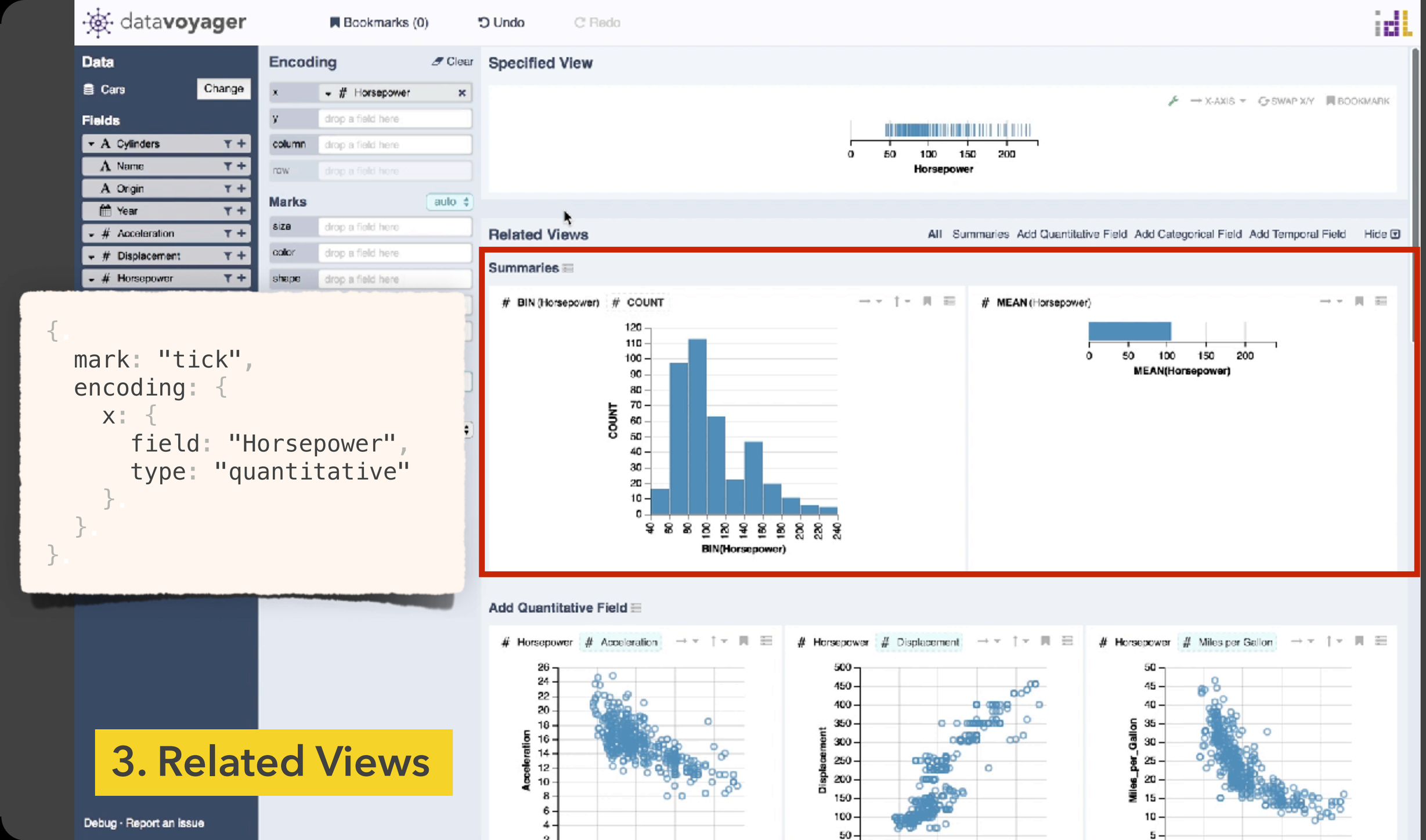


3. Related Views

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

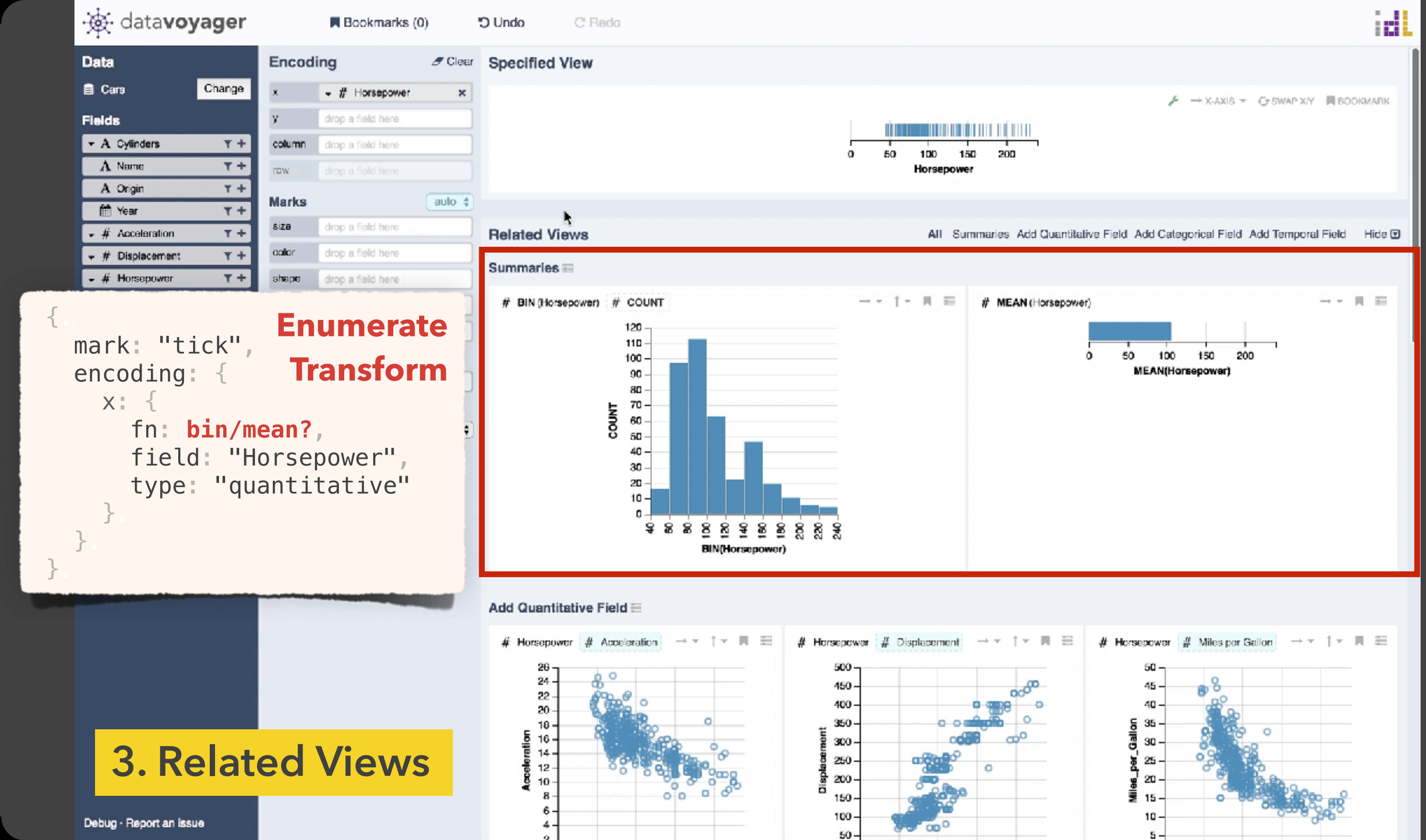


3. Related Views




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

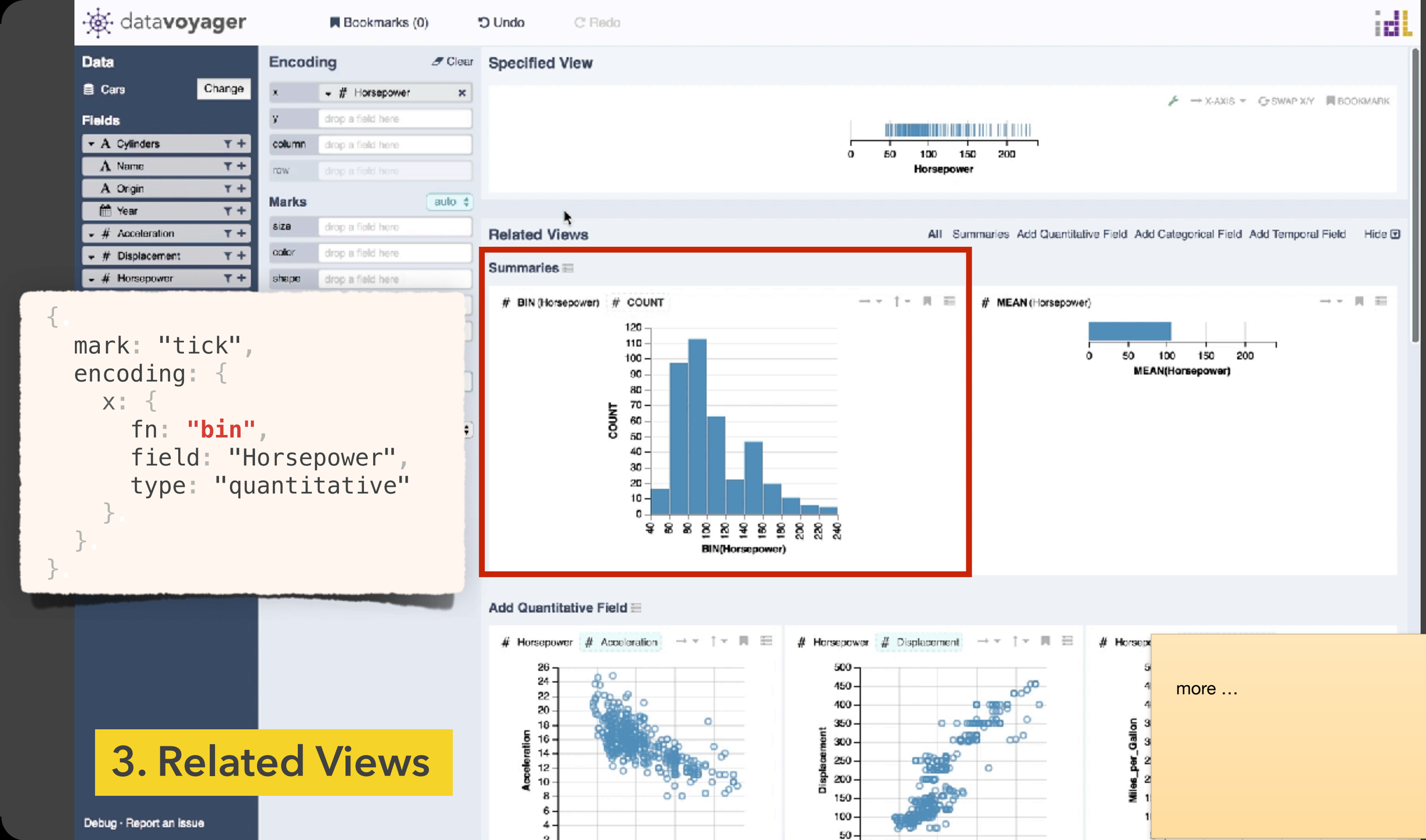
3. Related Views




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

Enumerate Transform

3. Related Views

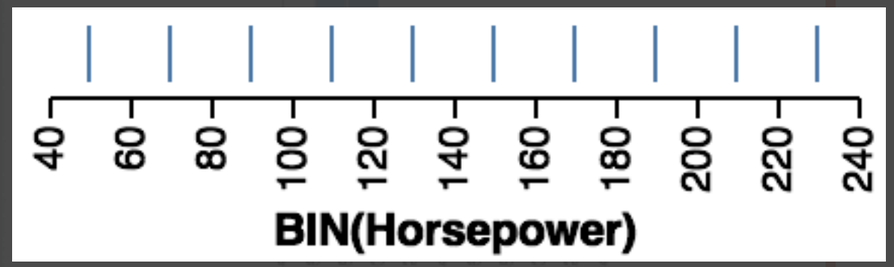



```
{
  mark: "tick",
  encoding: {
    x: {
      fn: "bin",
      field: "Horsepower",
      type: "quantitative"
    }
  }
}
```

more …

3. Related Views

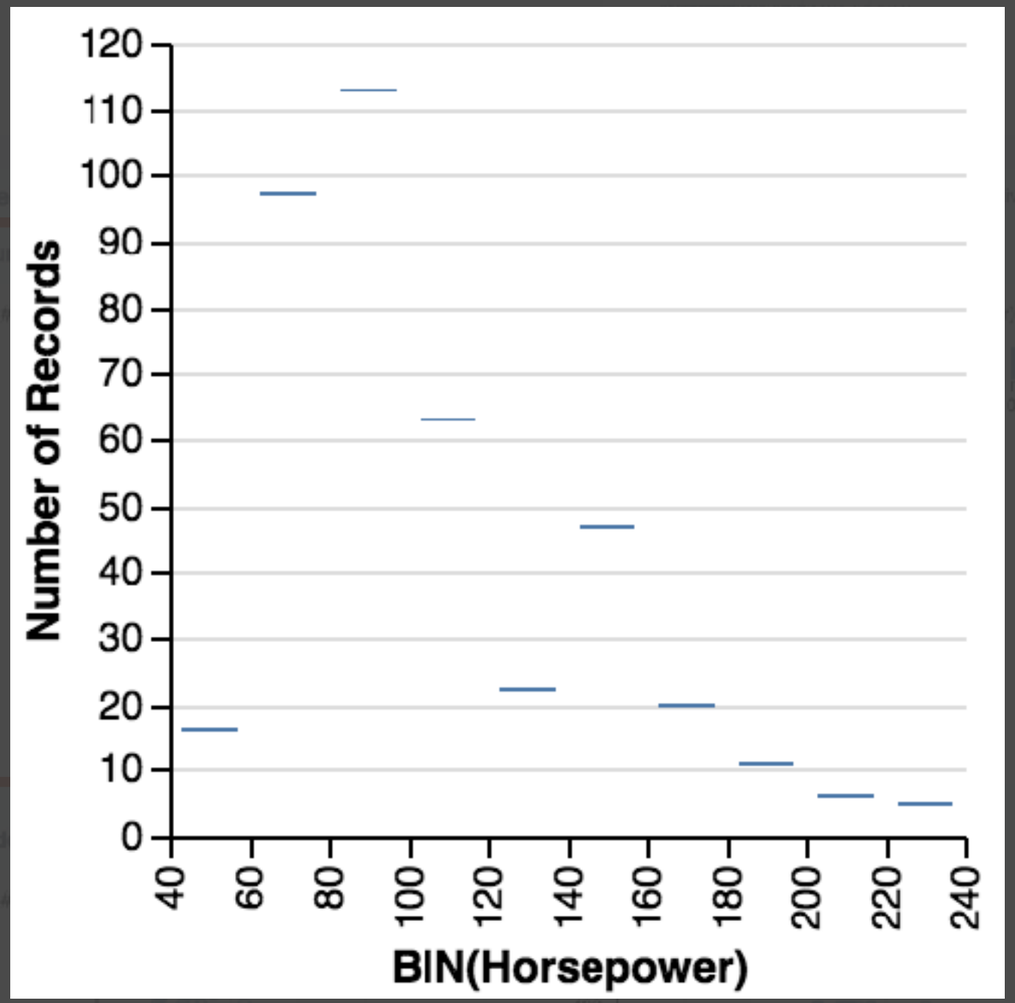
```
{
  mark: "tick",
  encoding: {
    x: {
      fn: "bin",
      field: "Horsepower",
      type: "quantitative"
    }
  }
}
```



more ...

3. Related Views

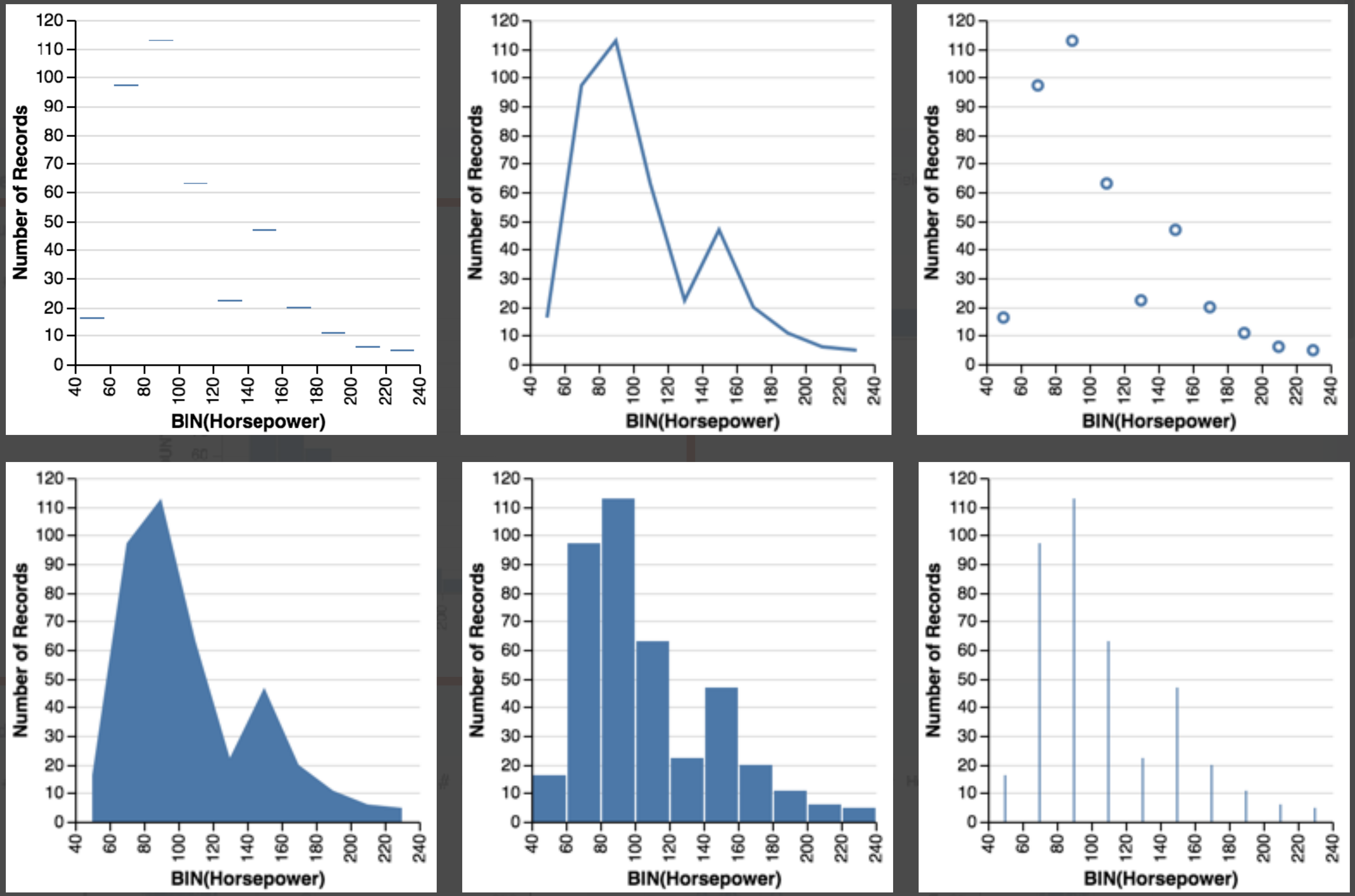
```
{
  mark: "tick",
  encoding: {
    x: {
      fn: "bin",
      field: "Horsepower",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
  }
}
```



3. Related Views

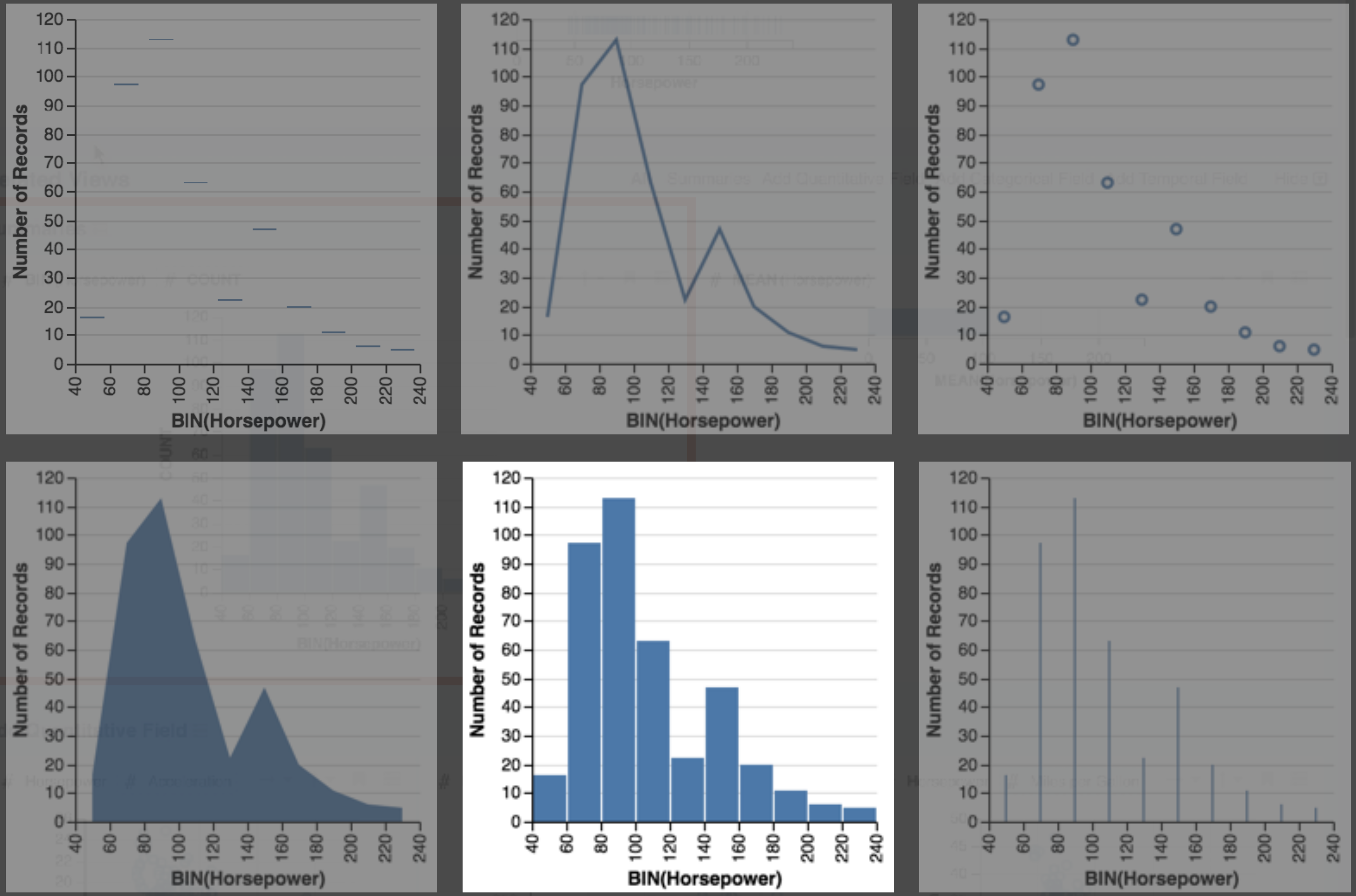
Enumerate & Rank Encodings

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

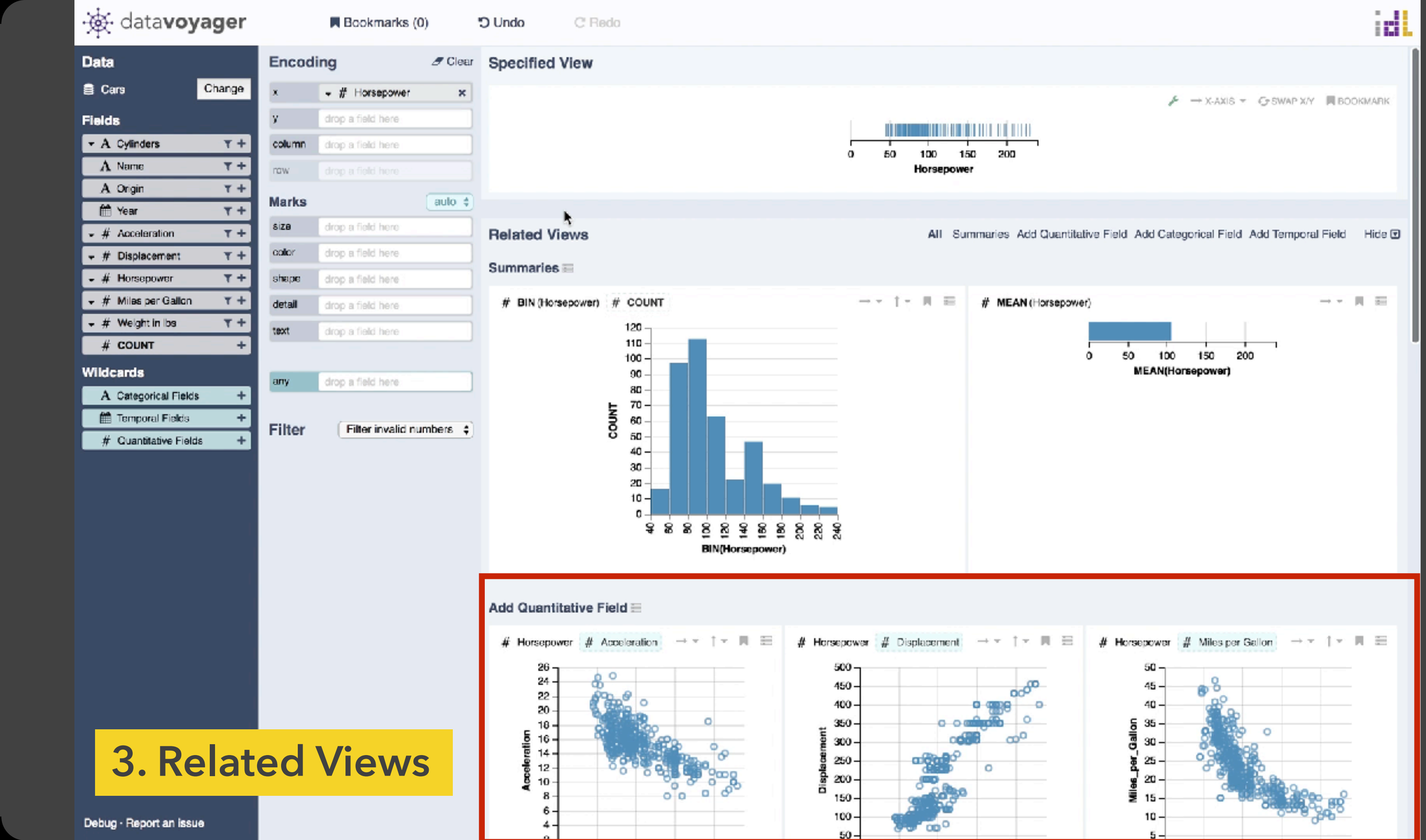


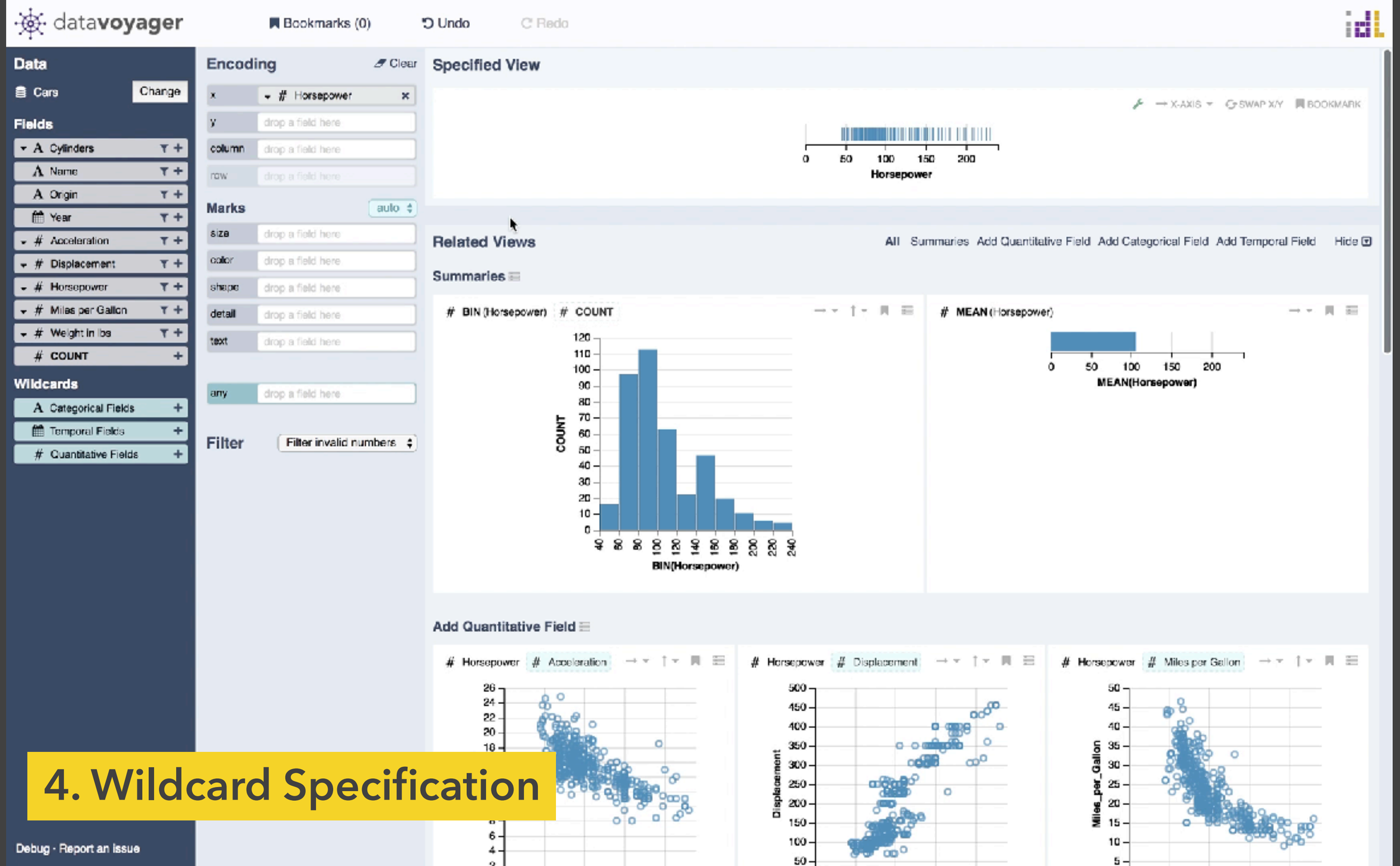
Enumerate & Rank Encodings

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

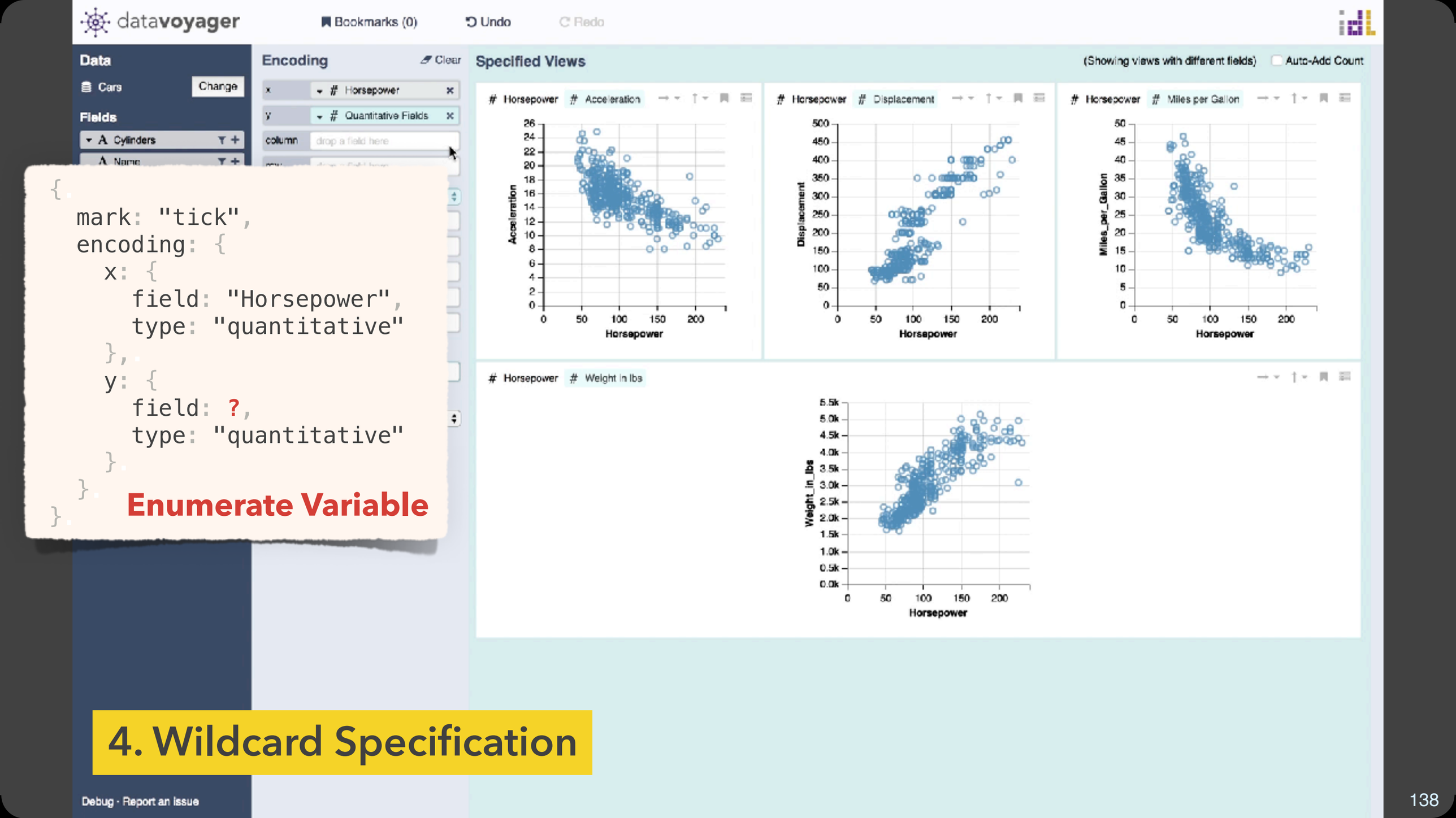


3. Related Views


datavoyager
Bookmarks (0)
Undo
Redo
Data
Cars
Change
Fields
Cylinders
Name
Origin
Year
Acceleration
Displacement
Horsepower
Miles per Gallon
Weight in lbs
COUNT
Wildcards
Categorical Fields
Temporal Fields
Quantitative Fields
Encoding
Clear
x
Horsepower
y
column
row
drop a field here
Marks
auto
size
color
shape
detail
text
any
Filter
Filter invalid numbers
Specified View
X-AXIS
SWAP X/Y
BOOKMARK
0
50
100
150
200
Horsepower
Related Views
All
Summaries
Add Quantitative Field
Add Categorical Field
Add Temporal Field
Hide
Summaries
BIN (Horsepower)
COUNT
BIN(Horsepower)
MEAN (Horsepower)
MEAN(Horsepower)
Add Quantitative Field
Acceleration
Displacement
Miles_per_Gallon
Debug · Report an issue
3. Related Views



4. Wildcard Specification


datavoyager
Bookmarks (0)
Undo
Redo
Data
Cars
Change
Fields
Cylinders
Encoding
Clear
x
Horsepower
y
Quantitative Fields
column
drop a field here
Specified Views
(Showing views with different fields)
Auto-Add Count
Horsepower
Acceleration
Displacement
Miles per Gallon
Weight in lbs
{
mark: "tick",
encoding: {
x: {
field: "Horsepower",
type: "quantitative"
},
y: {
field: ?,
type: "quantitative"
}
}
}
Enumerate Variable
4. Wildcard Specification
Debug · Report an issue

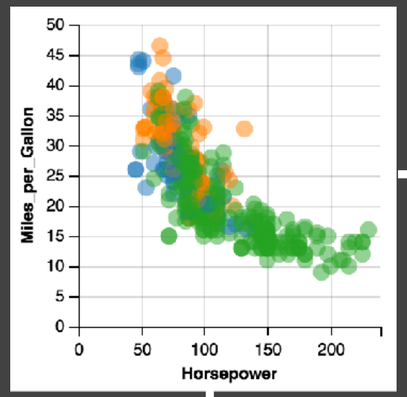
Voyager

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

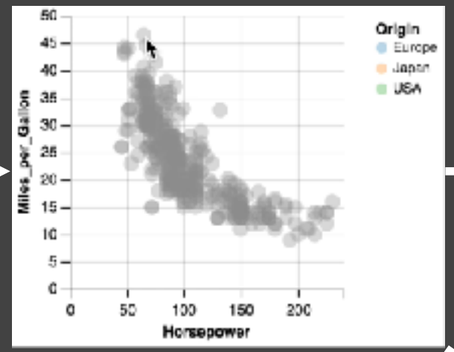
Use Vega-Lite to **recommend** data and visual encodings.

interactions?

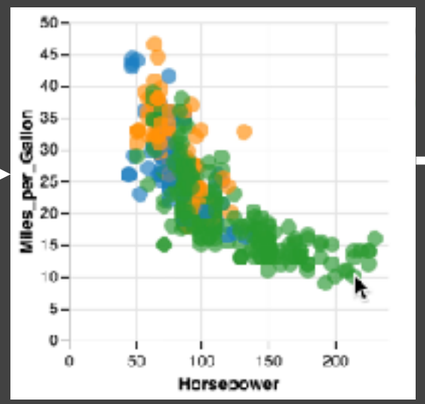
https://github.com/vega/voyager



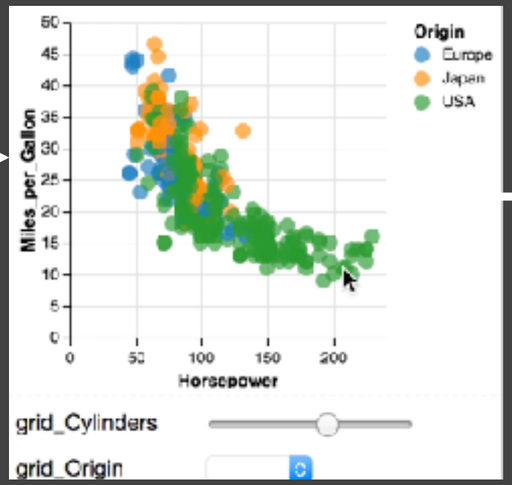
Single



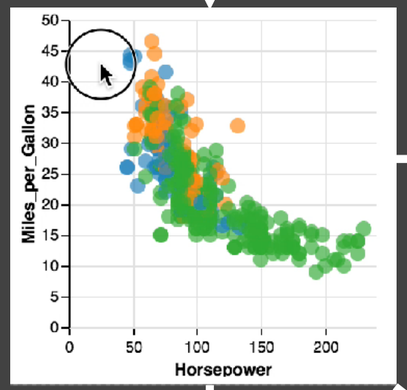
Project



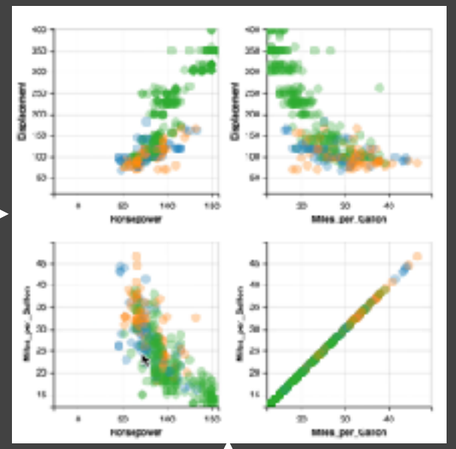
Bind



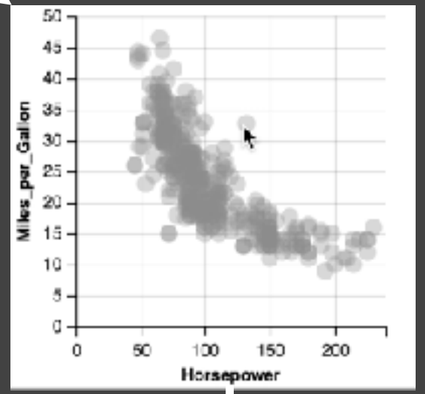
Interval



Repeat

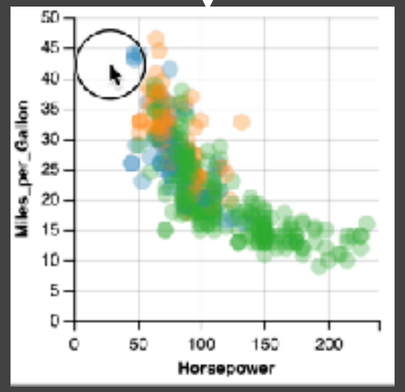


Multi



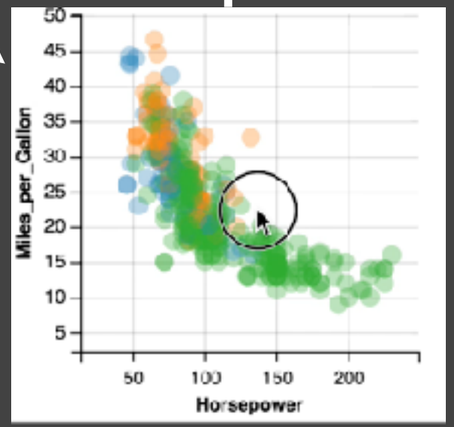
Systematically enumerate the **interaction design** space

Project

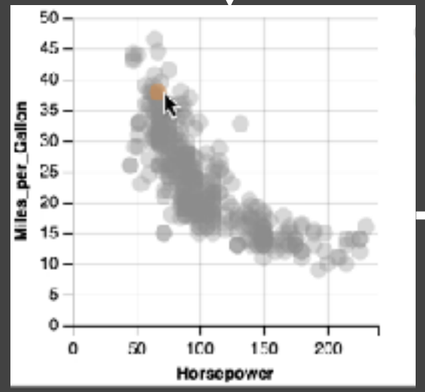


Bind

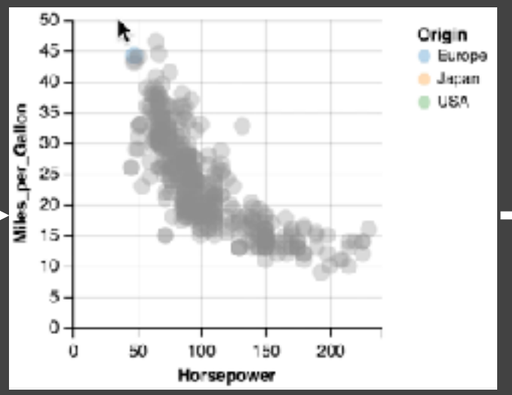
Repeat



Hover



Nearest



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

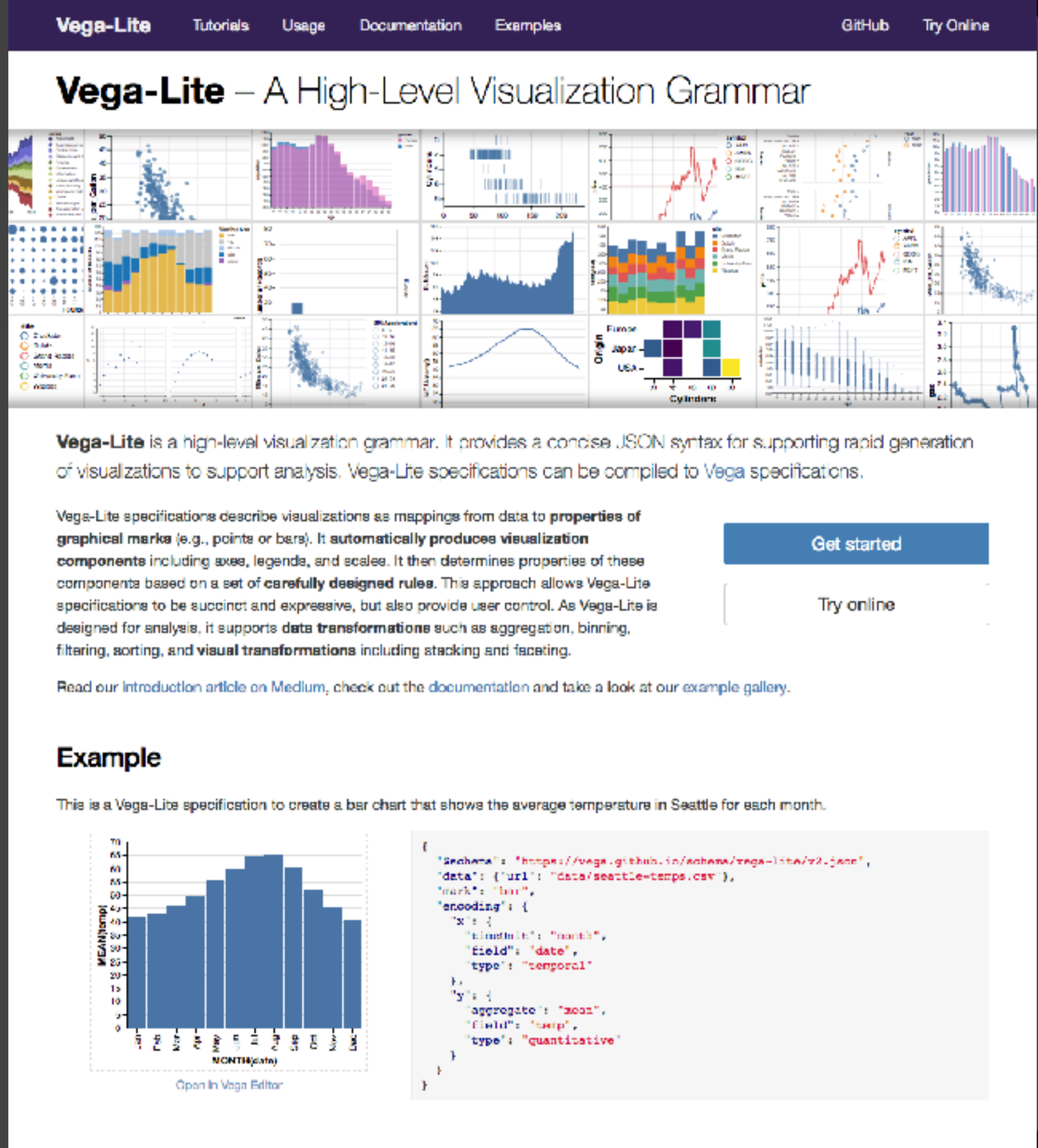
Higher-level **Tools** and **Recommendations**

Announcing

Vega-Lite 2 beta

(including multi-view composition and interaction support)

vega.github.io/vega-lite/



Exploring Data

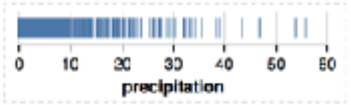
In this tutorial, you'll learn a few more techniques for creating visualizations in Vega-Lite. If you are not fa
Lite, please read the getting started tutorial first.

For this tutorial, we will create visualizations to explore weather data for Seattle, taken from NOAA. The d
file with columns for the temperature (in Celsius), precipitation (in centimeter), wind (in meter/second), an
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
specification.

```
"data": {"url": "data/seattle-weather.csv"}
```

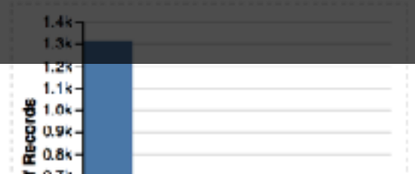
Let's start by looking at the precipitation. Precipitation is a quantitative variable. Let's use a tick mark to s
distribution of precipitation.



Open in Vega Editor

```
{
  "$schema": "https://vega.github.io/schema/vega-lit
  "data": {"url": "data/seattle-weather.csv"},
  "mark": "tick",
  "encoding": {
    "x": {"field": "precipitation", "type": "quantit
  }
}
```

It looks as though precipitation is skewed towards lower values; that is, when it rains, it usually doesn't r
better see this, we can create a histogram of the precipitation data. For this, we have to add an encoding
that uses a special field `*` that is aggregated with `count`. It is difficult to see patterns across continuou
can therefo[illegible] by [illegible]




```
{
  "$schema": "https://vega.github.io/schema/vega-lit
  "data": {"url": "data/seattle-weather.csv"},
  "mark": "bar",
  "encoding": {
    "x": {
```


Overview

Edit this page

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.

Table of Contents

Below is an overview of the documentation for Vega-Lite properties. See below for an overview of Vega-Lite specifications.

Tutorials & Documentations

vega.github.io/vega-lite

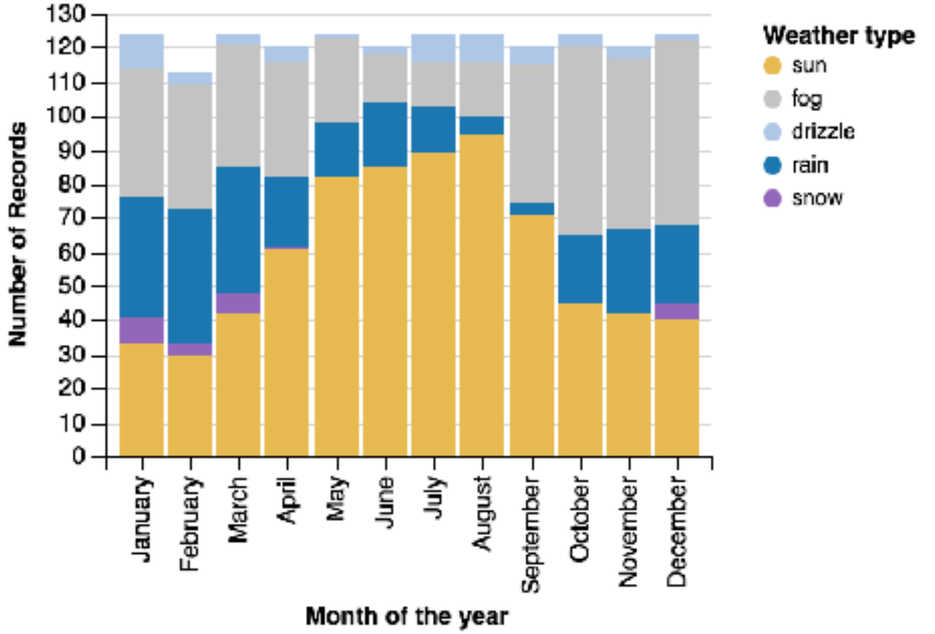
Vega-lite | Stacked Bar Chart | Format | Parse

```
{
  "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 compiled from Vega-Lite (read-only) | Edit Vega spec

```
{
  "width": 1,
  "height": 1,
  "padding": "auto",
  "data": [
    {
      "name": "source_0",
      "url": "data/seattle-weather.csv",
      "format": {"type": "csv","parse": {"date": "date"}},
      "transform": [
        {
          "type": "filter",
          "test": "datum[\"date\"] !== null && !isNaN(datum[\"date\"])"
```

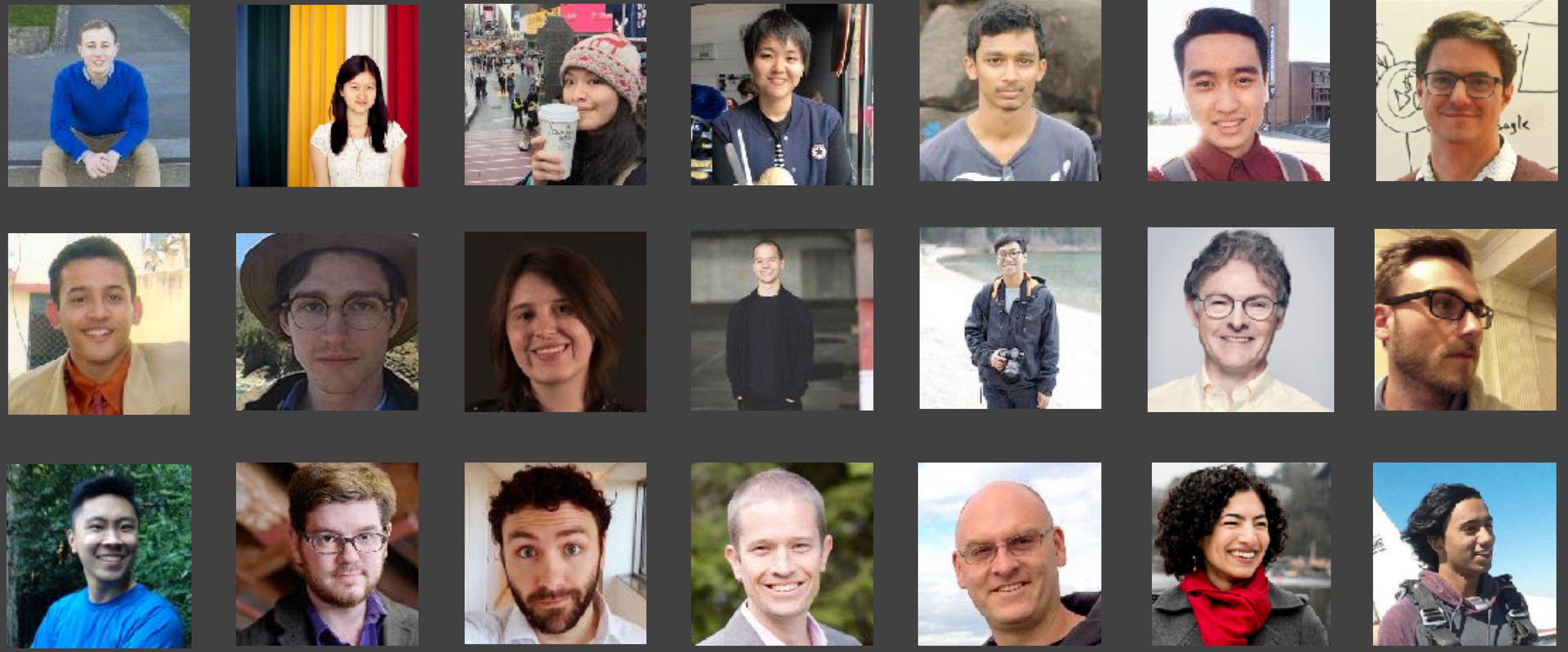
Visualization | Renderer: Canvas | Export



Online Editor

vega.github.io/new-editor

Contributors : Vega-Lite, Altair & Voyager



and many more…




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

Interactive Data Lab @uwdata
University of Washington