Structured, Declarative Data Visualization in Clojure

Clojure Remote
Feb. 11, 2016
Dave Tsao

dt@davetsao.com
github.com/dvdt

https://github.com/dvdt/gyptis

Problem: how to look at data in clojure?

R	ggplot2
Python	matplotlib
Clojure	???

Why visualize data?

In this talk:

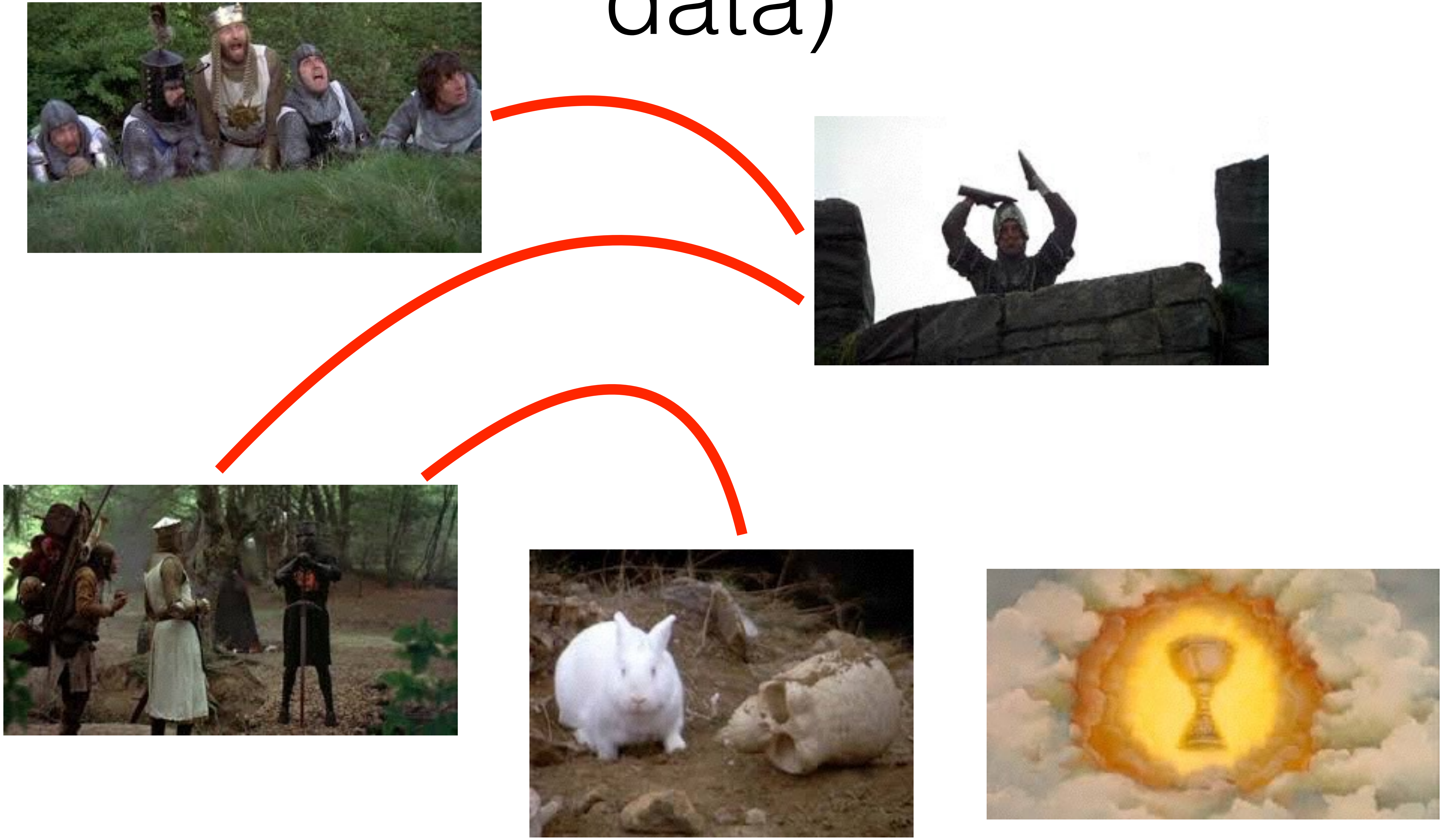
Test a hypothesis or answer a question

Communicate findings to others

Not in this talk:

Data wrangling / munging / reshaping

Quest for answers (from data)



Upon encountering data

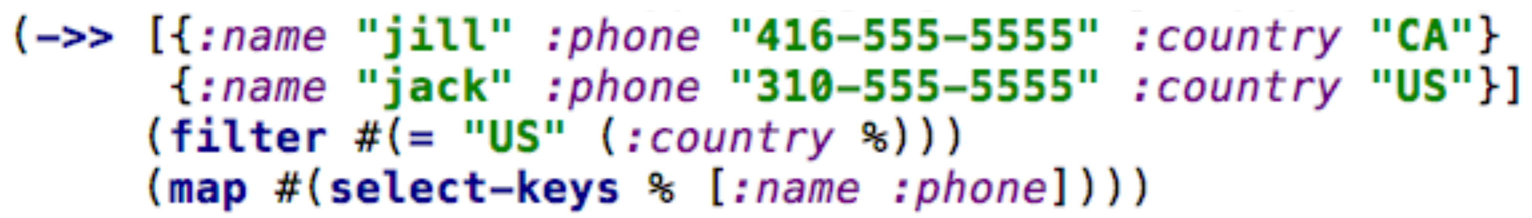
- Assumptions are shattered / refined
- More important questions are raised
- Pitfalls need to be avoided

Need rapid prototyping tool for data analysis

Interactive DSL for data vis.

- ‘Grammar of graphics’ == domain specific language for visualizing data.
- Implementations:
  - ggplot2 ‘**g**rammar of **g**raphics **plot**’
  - vega.js **<-** Gyptis wraps vega

What does this do?



```
(->> [{:name "jill" :phone "416-555-5555" :country "CA"}
      {:name "jack" :phone "310-555-5555" :country "US"}]
     (filter #(= "US" (:country %)))
     (map #(select-keys % [:name :phone])))
```

What does this do?

```clojure
(->> [{:name "jill" :phone "416-555-5555" :country "CA"}
      {:name "jack" :phone "310-555-5555" :country "US"}]
     (filter #(= "US" (:country %)))
     (map #(select-keys % [:name :phone])))
```

```sql
SELECT name, phone FROM users WHERE country='US';
```

Interactive: the REPL knows the answer

```clojure
(->> [{:name "jill" :phone "416-555-5555" :country "CA"}
      {:name "jack" :phone "310-555-5555" :country "US"}]
     (filter #(= "US" (:country %)))
     (map #(select-keys % [:name :phone])))
```

```sql
SELECT name, phone FROM users WHERE country='US';
```

```
=> ({:name "jack", :phone "310-555-5555"})
```

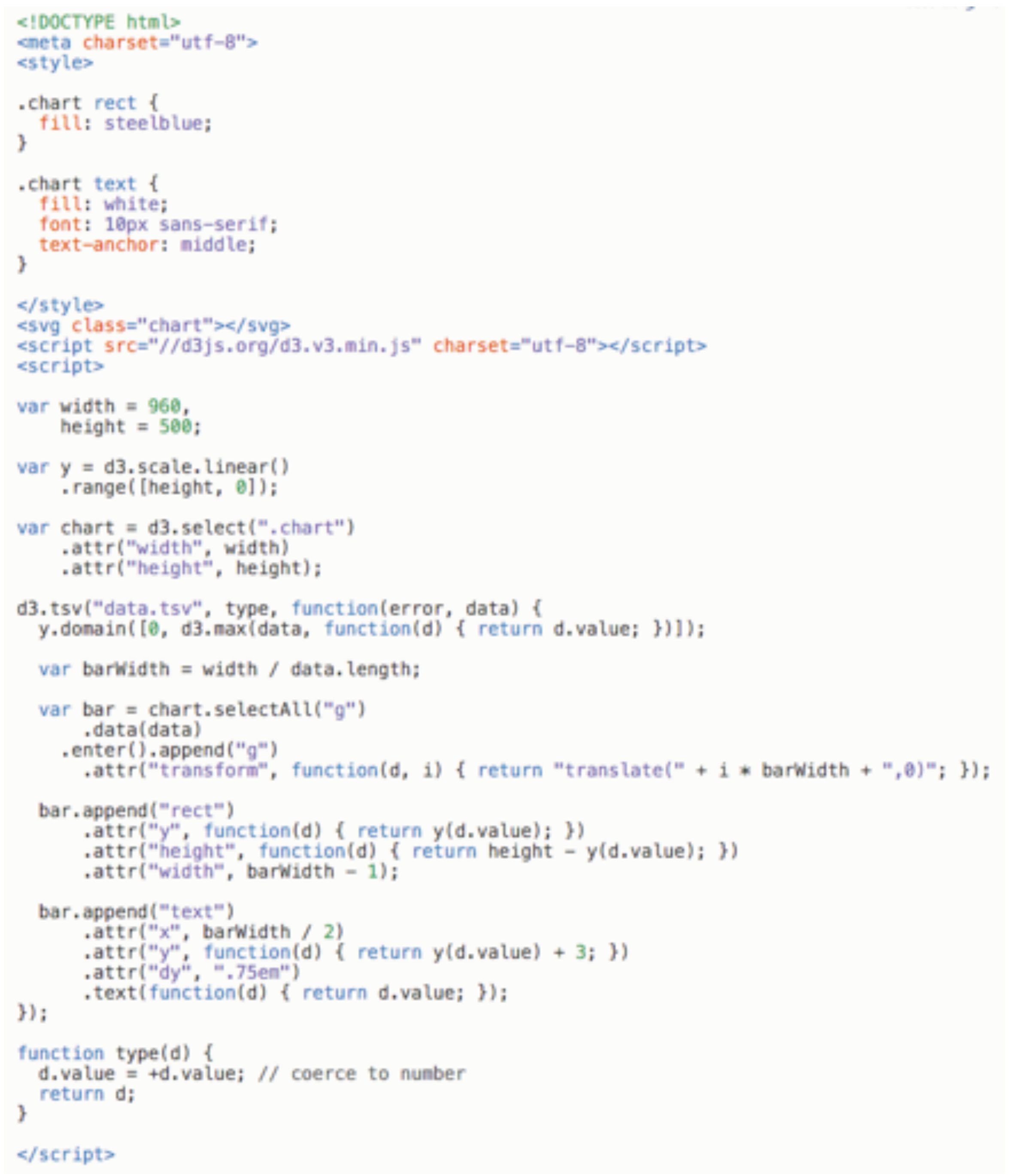
Technology hierarchy

gyptis Plotting for data analysis

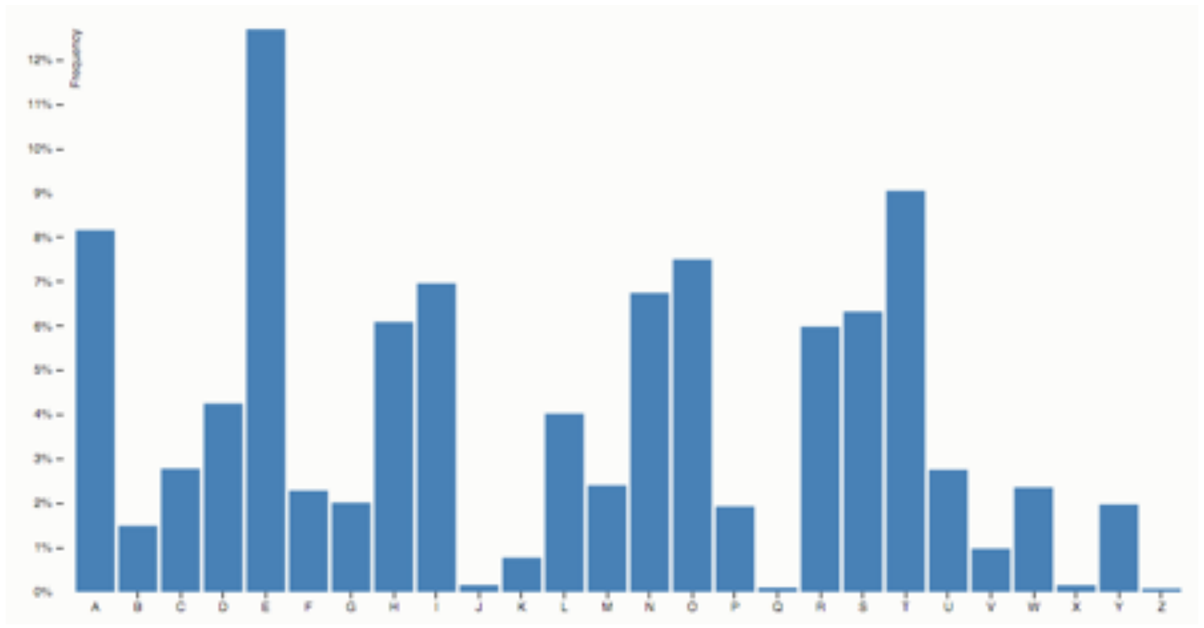
vega A visualization grammar

D3 Data-Driven Documents

D3: Let’s make a bar chart

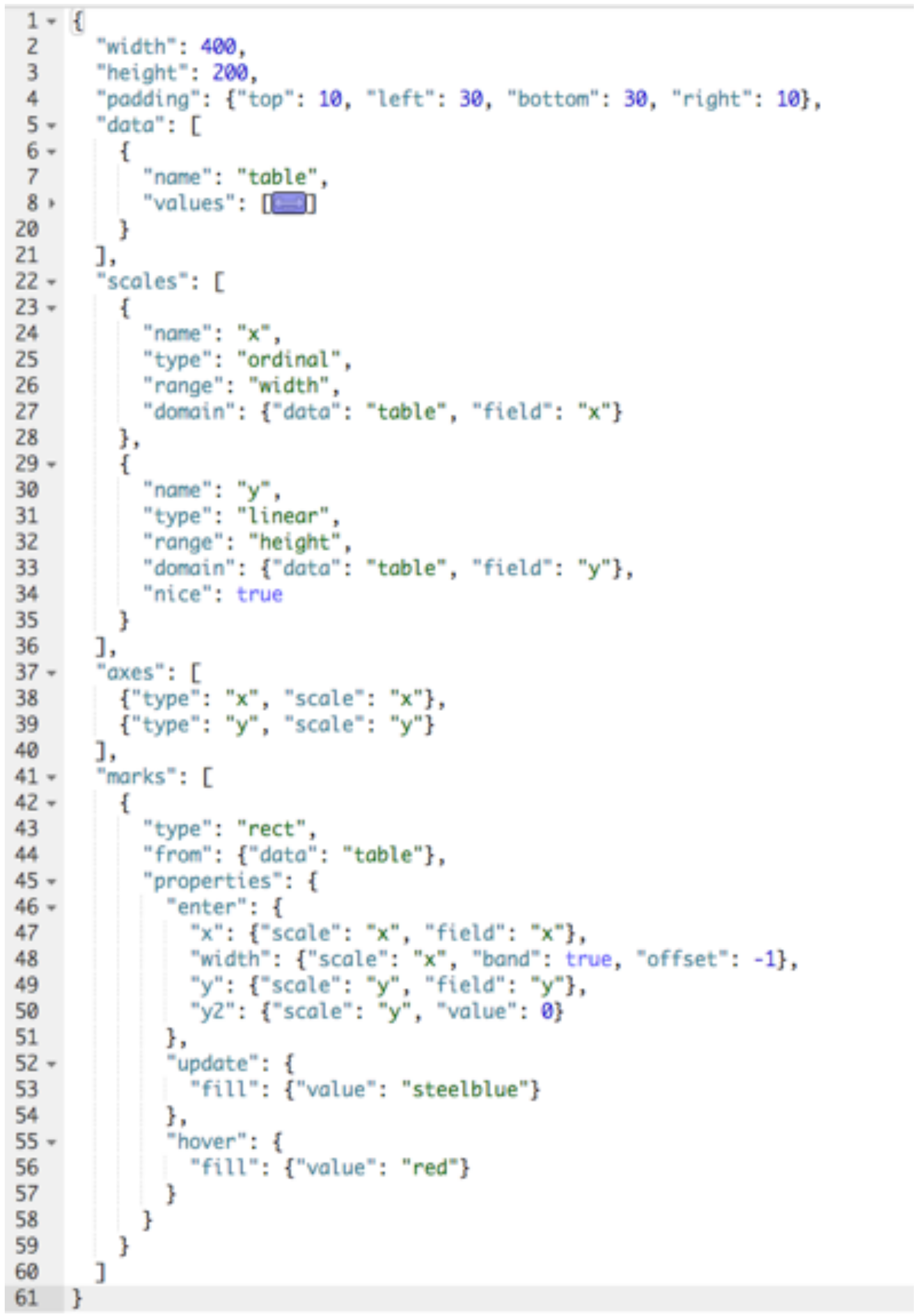


```
<!DOCTYPE html>
<meta charset="utf-8">
<style>

.chart rect {
  fill: steelblue;
}

.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: middle;
}

</style>
<svg class="chart"></svg>
<script src="//d3js.org/d3.v3.min.js" charset="utf-8"></script>
<script>

var width = 960,
    height = 500;

var y = d3.scale.linear()
    .range([height, 0]);

var chart = d3.select(".chart")
    .attr("width", width)
    .attr("height", height);

d3.tsv("data.tsv", type, function(error, data) {
  y.domain([0, d3.max(data, function(d) { return d.value; })]);

  var barWidth = width / data.length;

  var bar = chart.selectAll("g")
      .data(data)
    .enter().append("g")
      .attr("transform", function(d, i) { return "translate(" + i * barWidth + ",0)"; });

  bar.append("rect")
      .attr("y", function(d) { return y(d.value); })
      .attr("height", function(d) { return height - y(d.value); })
      .attr("width", barWidth - 1);

  bar.append("text")
      .attr("x", barWidth / 2)
      .attr("y", function(d) { return y(d.value) + 3; })
      .attr("dy", ".75em")
      .text(function(d) { return d.value; });
});

function type(d) {
  d.value = +d.value; // coerce to number
  return d;
}

</script>
```

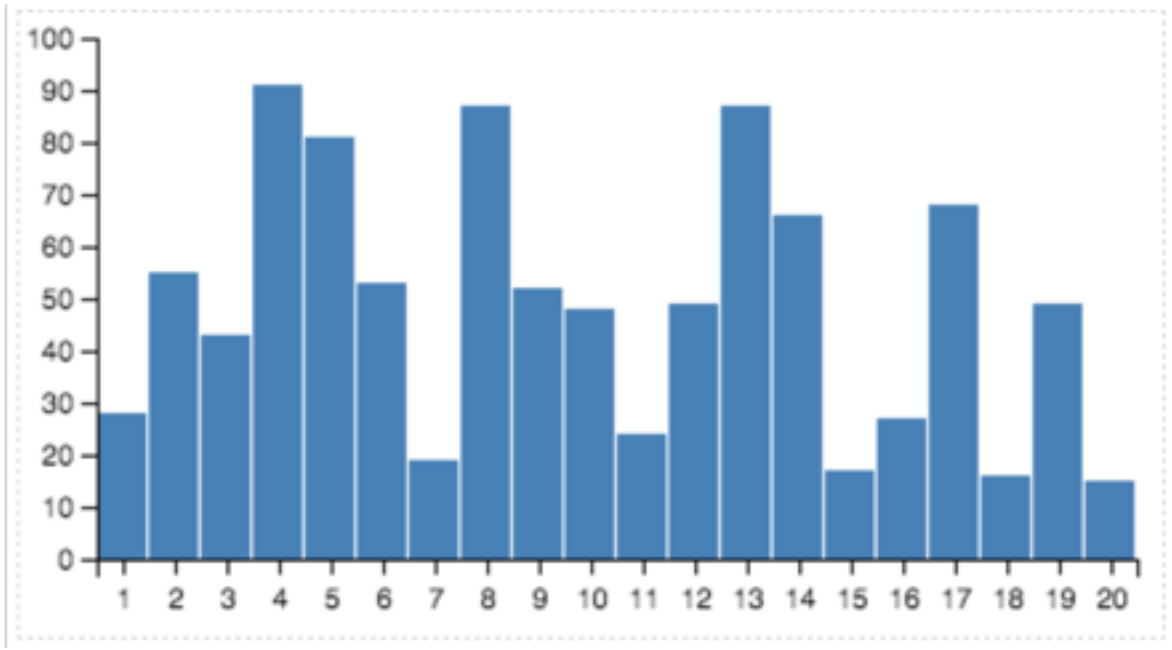


http://bost.ocks.org/mike/bar/3/

Vega: Let’s make a bar chart



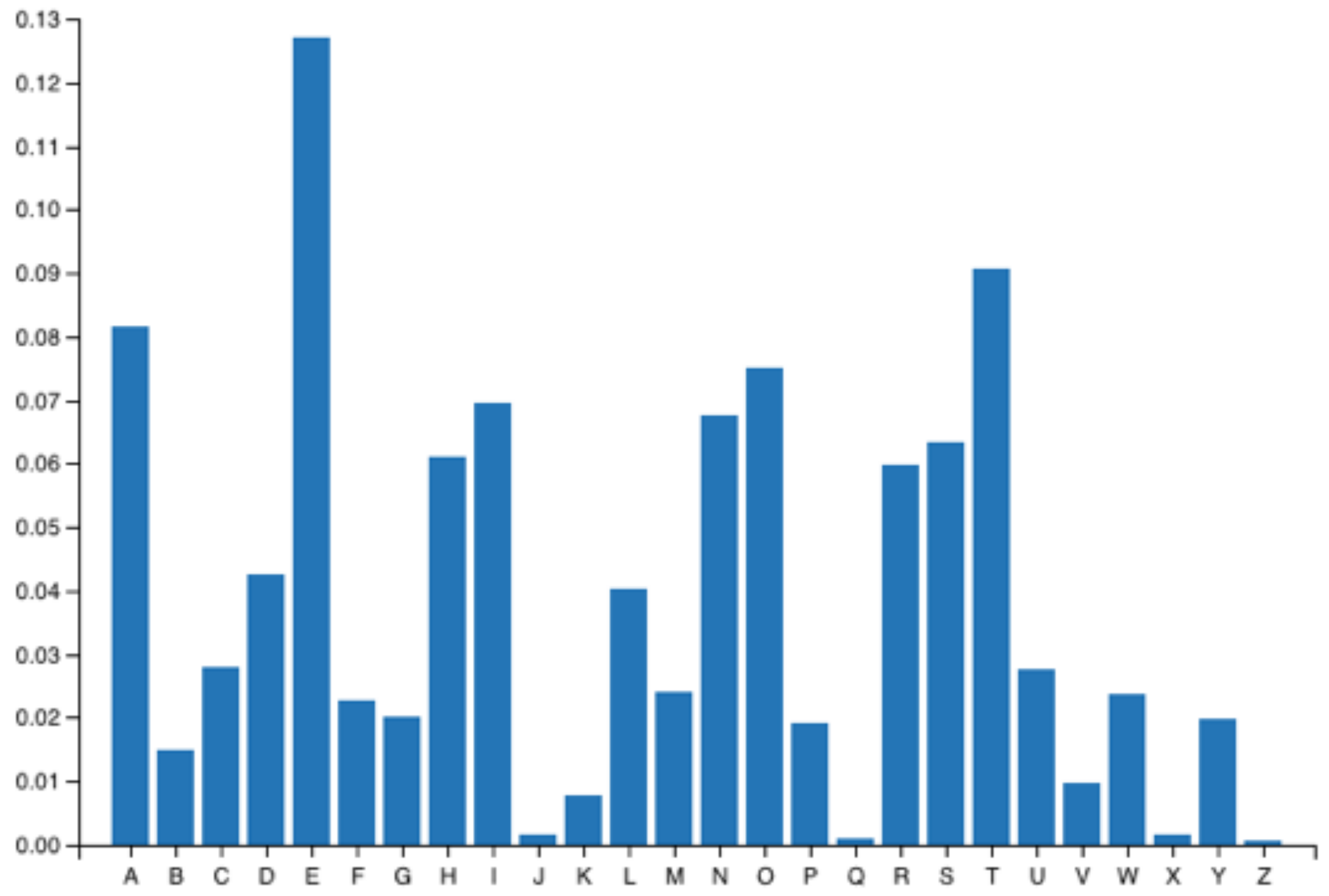
```
{
  "width": 400,
  "height": 200,
  "padding": {"top": 10, "left": 30, "bottom": 30, "right": 10},
  "data": [
    {
      "name": "table",
      "values": [↔]
    }
  ],
  "scales": [
    {
      "name": "x",
      "type": "ordinal",
      "range": "width",
      "domain": {"data": "table", "field": "x"}
    },
    {
      "name": "y",
      "type": "linear",
      "range": "height",
      "domain": {"data": "table", "field": "y"},
      "nice": true
    }
  ],
  "axes": [
    {"type": "x", "scale": "x"},
    {"type": "y", "scale": "y"}
  ],
  "marks": [
    {
      "type": "rect",
      "from": {"data": "table"},
      "properties": {
        "enter": {
          "x": {"scale": "x", "field": "x"},
          "width": {"scale": "x", "band": true, "offset": -1},
          "y": {"scale": "y", "field": "y"},
          "y2": {"scale": "y", "value": 0}
        },
        "update": {
          "fill": {"value": "steelblue"}
        },
        "hover": {
          "fill": {"value": "red"}
        }
      }
    }
  ]
}
```



http://vega.github.io/vega-editor/index.html?mode=vega

Gyptis: Let’s make a bar chart

```
(def letter-plot
  (gyptis/dodged-bar letter-frequency {:x :letter, :y :frequency}))

(view/plot! (-> letter-plot
                (assoc :height 400)
                (assoc :width 600)))
```



Technology hierarchy

gyptis	Plotting for data analysis	*Data*
vega	A visualization grammar	*Graphics primitives*
D3 Data-Driven Documents		*DOM Manipulation*

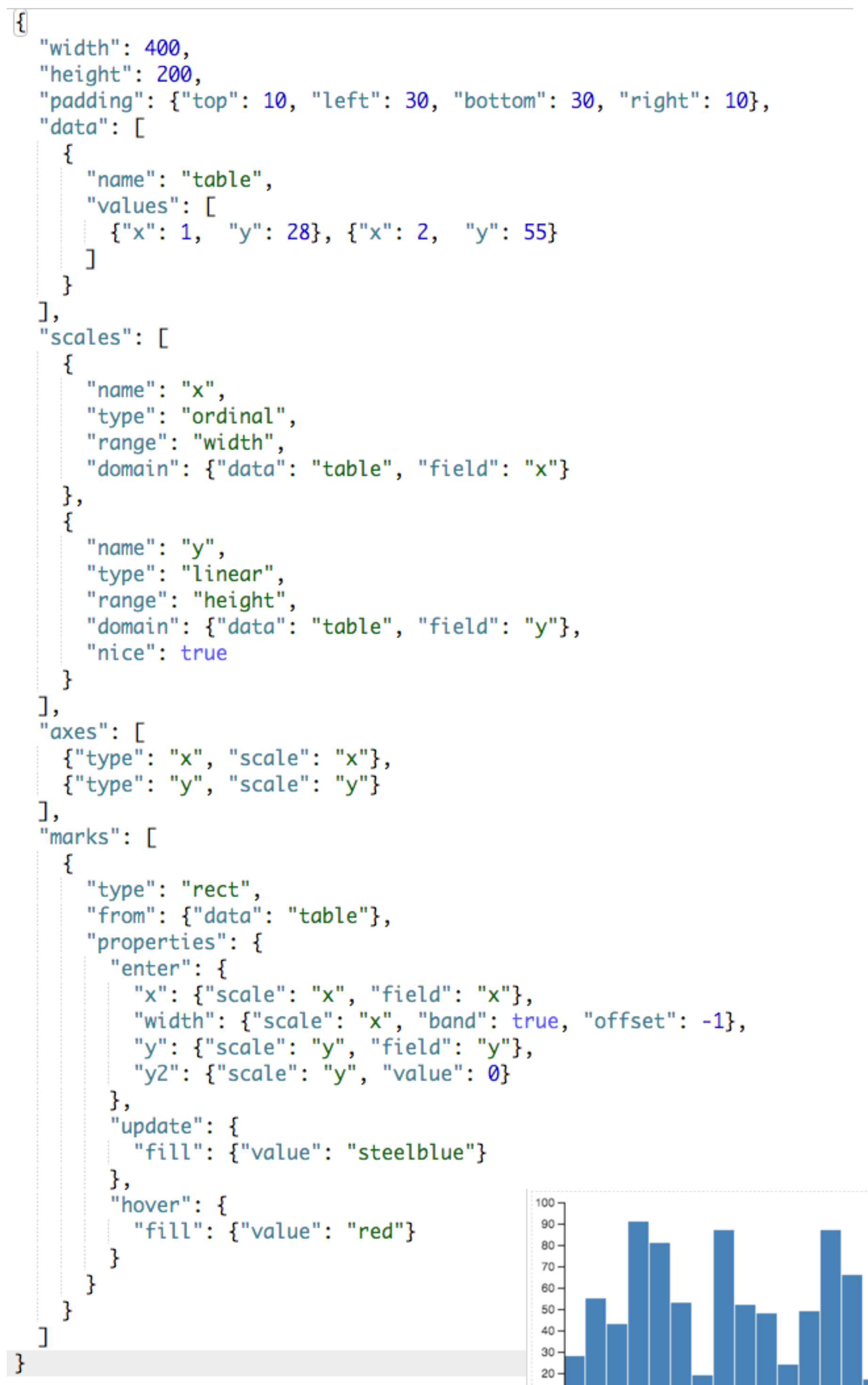
Gyptis: a clojure library for generating and viewing Vega plots

- Gyptis is:
    - a collection pure functions for generating vega plot specifications.
    - Websocket server for rendering vega plots on the browser.
- A vega plot is: JSON encoded specification of **web** graphics.

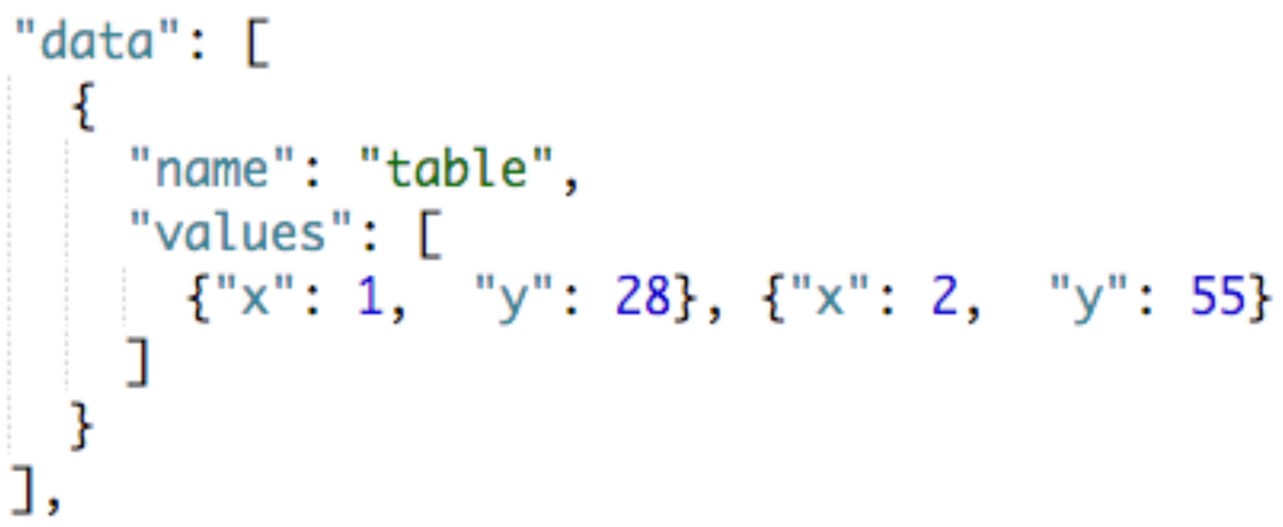
The Vega grammar

- **Marks** are visual elements of data (i.e. rectangle, circle, line)
- **Scales** transform data coordinate system (%, $, CPC) to display coordinates (pixels)
- **Axes** and **Legends** are the inverse of scales.

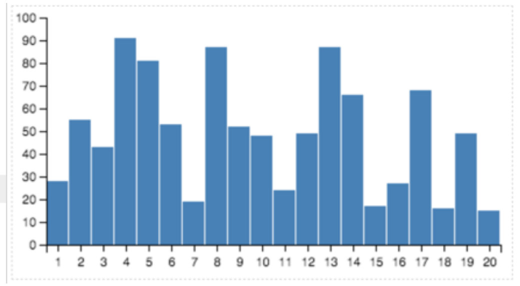
The Vega grammar



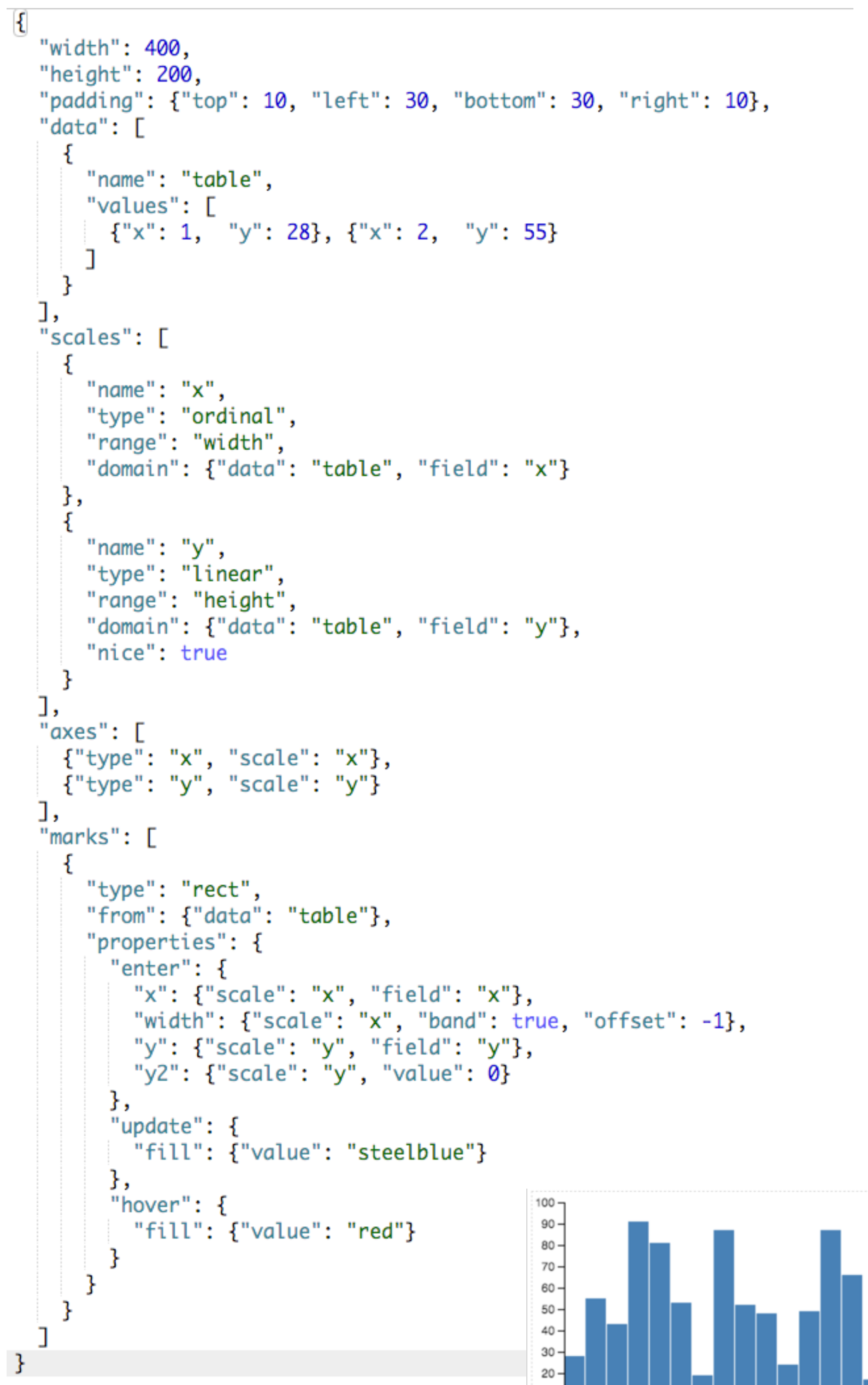
```
{
  "width": 400,
  "height": 200,
  "padding": {"top": 10, "left": 30, "bottom": 30, "right": 10},
  "data": [
    {
      "name": "table",
      "values": [
        {"x": 1,  "y": 28}, {"x": 2,  "y": 55}
      ]
    }
  ],
  "scales": [
    {
      "name": "x",
      "type": "ordinal",
      "range": "width",
      "domain": {"data": "table", "field": "x"}
    },
    {
      "name": "y",
      "type": "linear",
      "range": "height",
      "domain": {"data": "table", "field": "y"},
      "nice": true
    }
  ],
  "axes": [
    {"type": "x", "scale": "x"},
    {"type": "y", "scale": "y"}
  ],
  "marks": [
    {
      "type": "rect",
      "from": {"data": "table"},
      "properties": {
        "enter": {
          "x": {"scale": "x", "field": "x"},
          "width": {"scale": "x", "band": true, "offset": -1},
          "y": {"scale": "y", "field": "y"},
          "y2": {"scale": "y", "value": 0}
        },
        "update": {
          "fill": {"value": "steelblue"}
        },
        "hover": {
          "fill": {"value": "red"}
        }
      }
    }
  ]
}
```



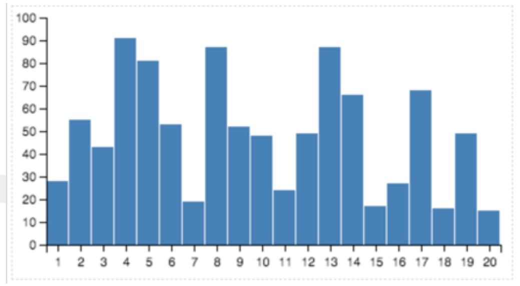
```
"data": [
  {
    "name": "table",
    "values": [
      {"x": 1,  "y": 28}, {"x": 2,  "y": 55}
    ]
  }
],
```

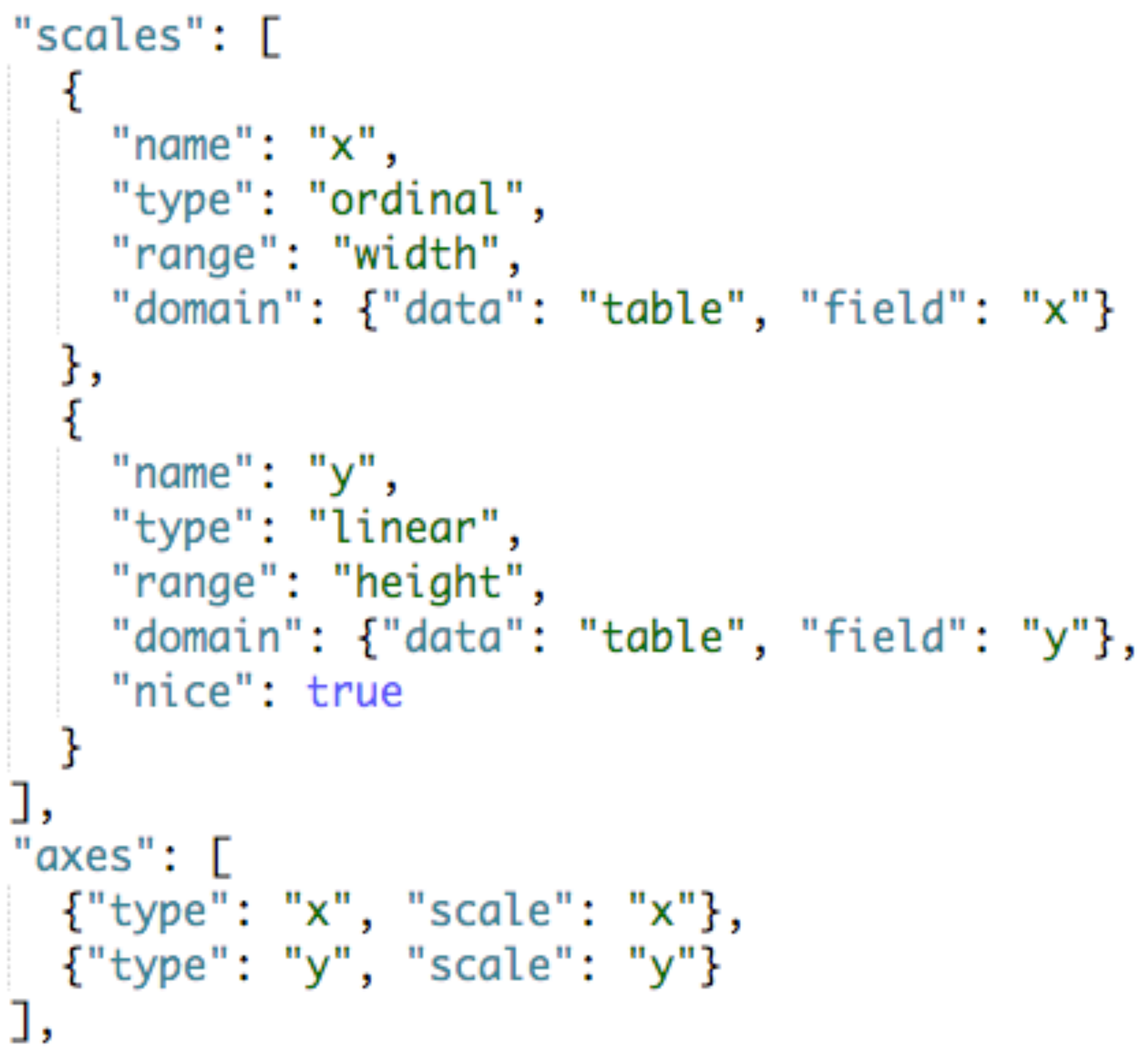


The Vega grammar



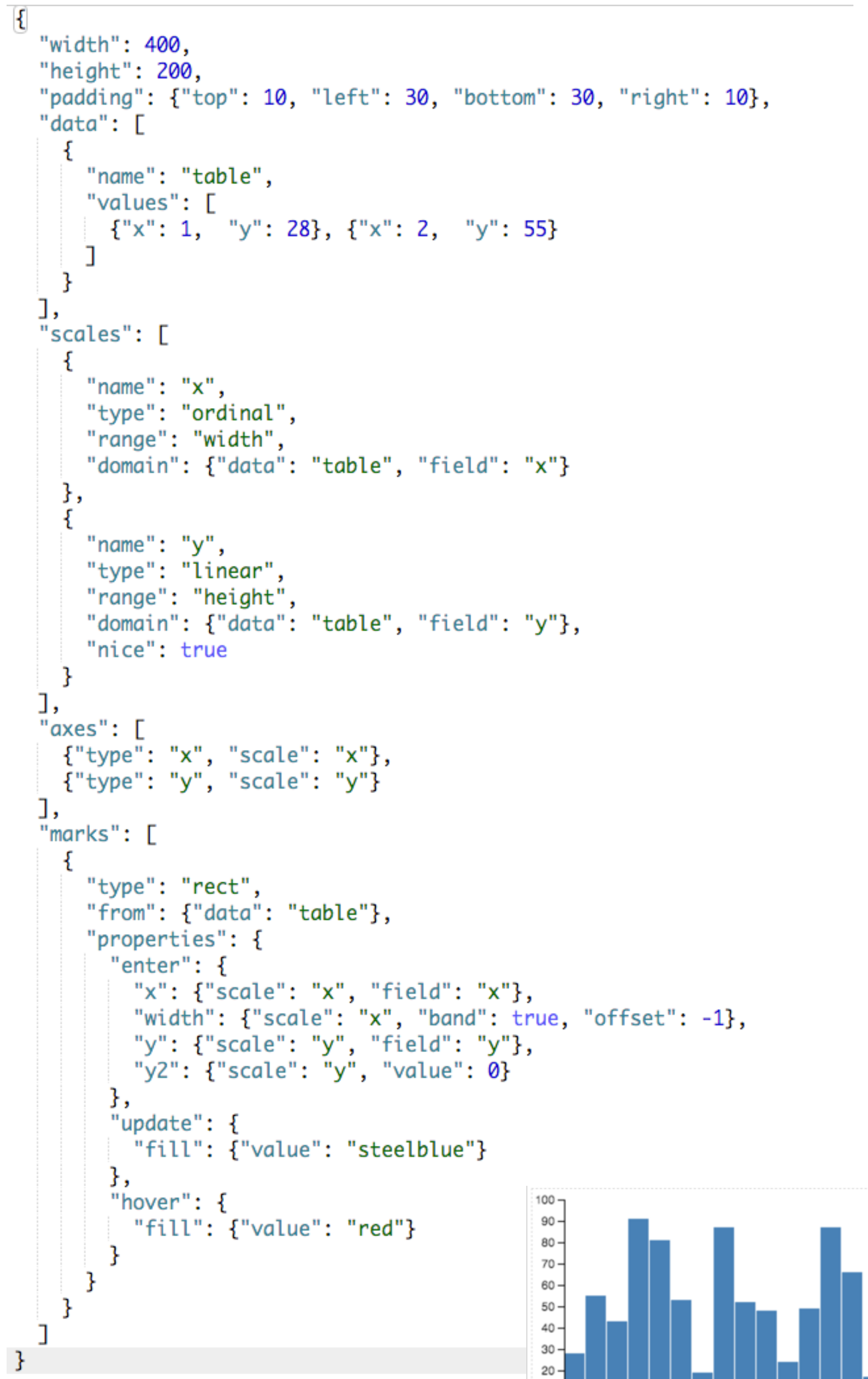
```
{
  "width": 400,
  "height": 200,
  "padding": {"top": 10, "left": 30, "bottom": 30, "right": 10},
  "data": [
    {
      "name": "table",
      "values": [
        {"x": 1,  "y": 28}, {"x": 2,  "y": 55}
      ]
    }
  ],
  "scales": [
    {
      "name": "x",
      "type": "ordinal",
      "range": "width",
      "domain": {"data": "table", "field": "x"}
    },
    {
      "name": "y",
      "type": "linear",
      "range": "height",
      "domain": {"data": "table", "field": "y"},
      "nice": true
    }
  ],
  "axes": [
    {"type": "x", "scale": "x"},
    {"type": "y", "scale": "y"}
  ],
  "marks": [
    {
      "type": "rect",
      "from": {"data": "table"},
      "properties": {
        "enter": {
          "x": {"scale": "x", "field": "x"},
          "width": {"scale": "x", "band": true, "offset": -1},
          "y": {"scale": "y", "field": "y"},
          "y2": {"scale": "y", "value": 0}
        },
        "update": {
          "fill": {"value": "steelblue"}
        },
        "hover": {
          "fill": {"value": "red"}
        }
      }
    }
  ]
}
```



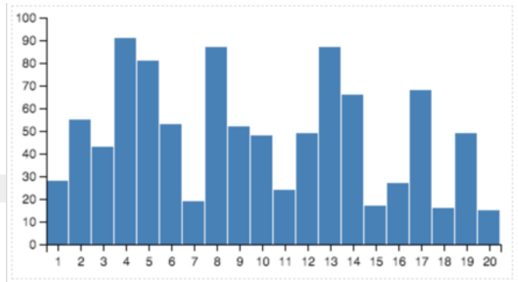


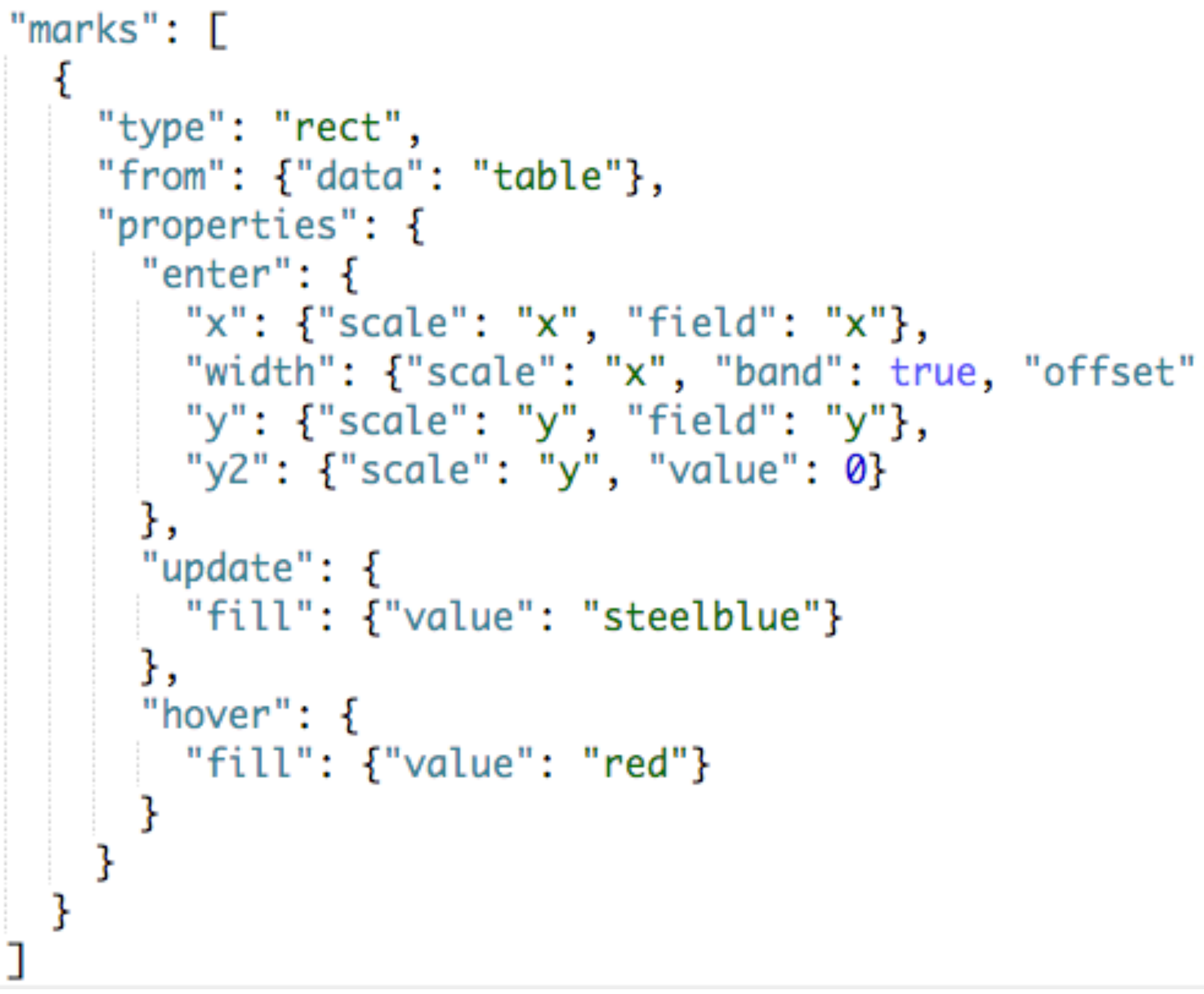
```
"scales": [
  {
    "name": "x",
    "type": "ordinal",
    "range": "width",
    "domain": {"data": "table", "field": "x"}
  },
  {
    "name": "y",
    "type": "linear",
    "range": "height",
    "domain": {"data": "table", "field": "y"},
    "nice": true
  }
],
"axes": [
  {"type": "x", "scale": "x"},
  {"type": "y", "scale": "y"}
],
```

The Vega grammar



```
{
  "width": 400,
  "height": 200,
  "padding": {"top": 10, "left": 30, "bottom": 30, "right": 10},
  "data": [
    {
      "name": "table",
      "values": [
        {"x": 1,  "y": 28}, {"x": 2,  "y": 55}
      ]
    }
  ],
  "scales": [
    {
      "name": "x",
      "type": "ordinal",
      "range": "width",
      "domain": {"data": "table", "field": "x"}
    },
    {
      "name": "y",
      "type": "linear",
      "range": "height",
      "domain": {"data": "table", "field": "y"},
      "nice": true
    }
  ],
  "axes": [
    {"type": "x", "scale": "x"},
    {"type": "y", "scale": "y"}
  ],
  "marks": [
    {
      "type": "rect",
      "from": {"data": "table"},
      "properties": {
        "enter": {
          "x": {"scale": "x", "field": "x"},
          "width": {"scale": "x", "band": true, "offset": -1},
          "y": {"scale": "y", "field": "y"},
          "y2": {"scale": "y", "value": 0}
        },
        "update": {
          "fill": {"value": "steelblue"}
        },
        "hover": {
          "fill": {"value": "red"}
        }
      }
    }
  ]
}
```



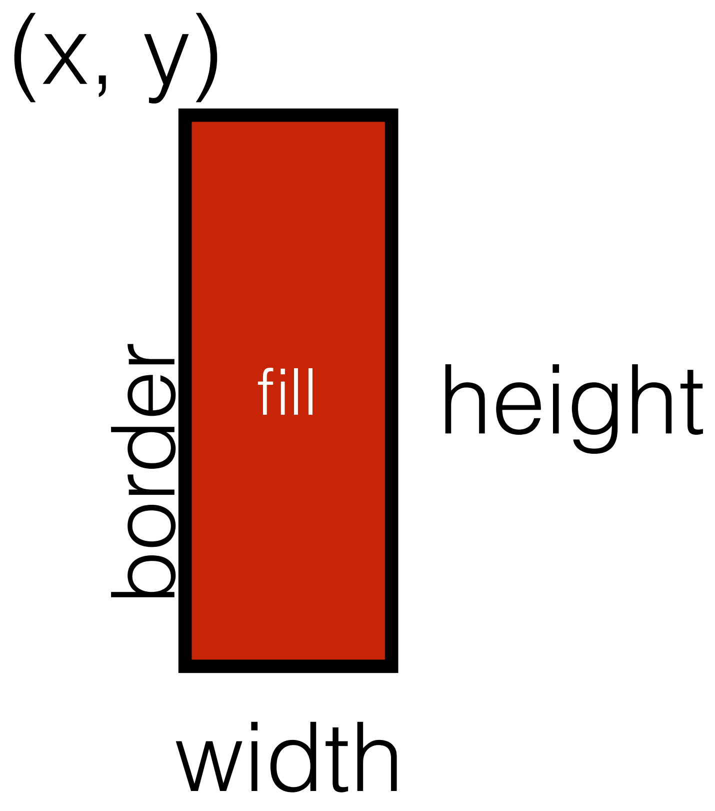


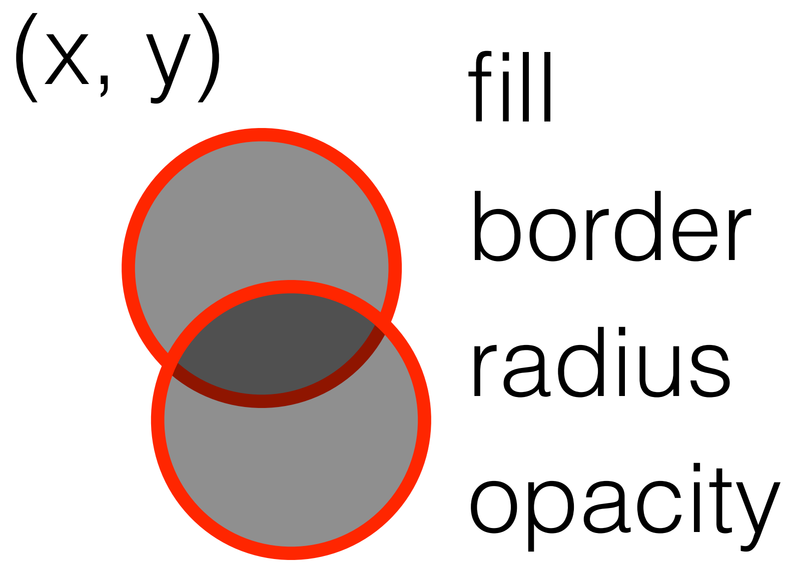
```
"marks": [
  {
    "type": "rect",
    "from": {"data": "table"},
    "properties": {
      "enter": {
        "x": {"scale": "x", "field": "x"},
        "width": {"scale": "x", "band": true, "offset"
        "y": {"scale": "y", "field": "y"},
        "y2": {"scale": "y", "value": 0}
      },
      "update": {
        "fill": {"value": "steelblue"}
      },
      "hover": {
        "fill": {"value": "red"}
      }
    }
  }
]
```

Demo

https://github.com/dvdt/gyptis-cljremote

Columns to aesthetics

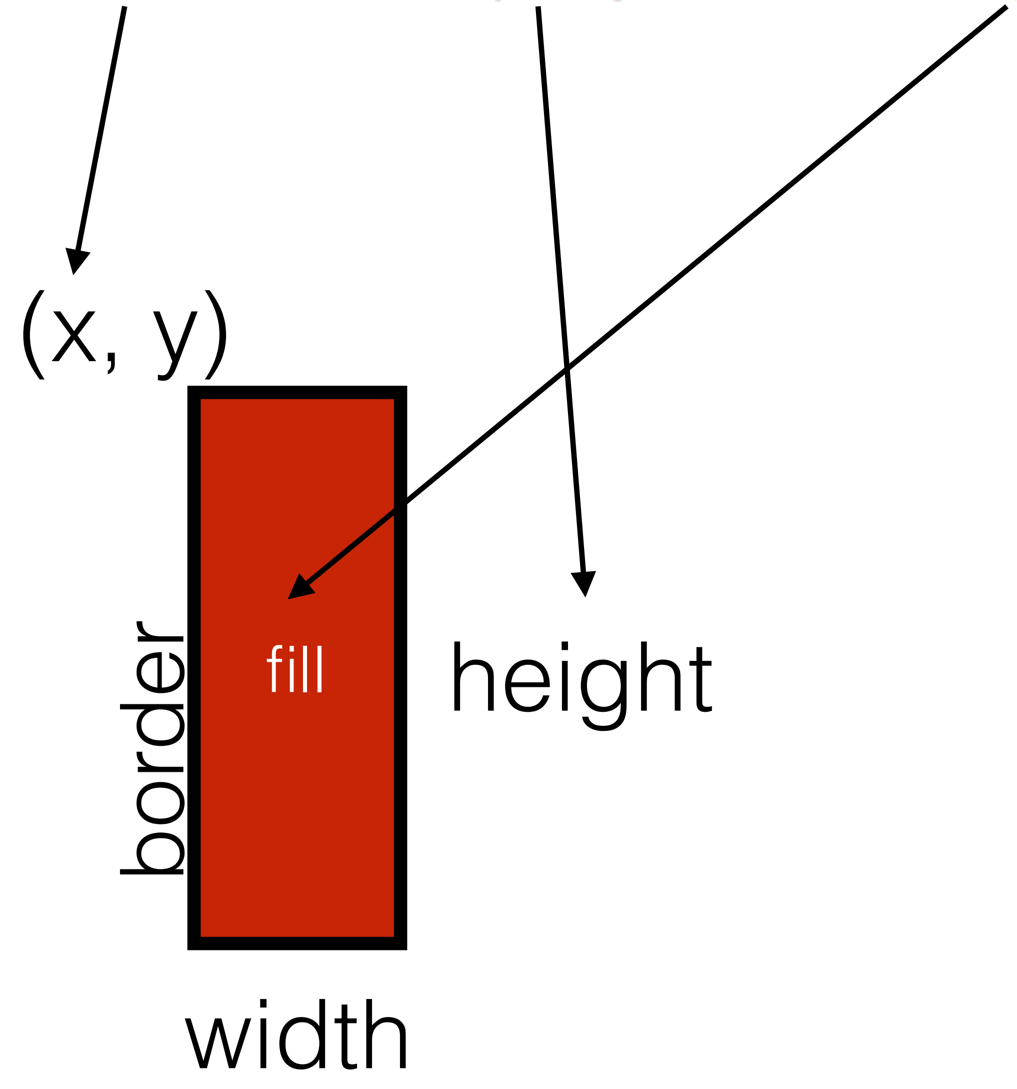




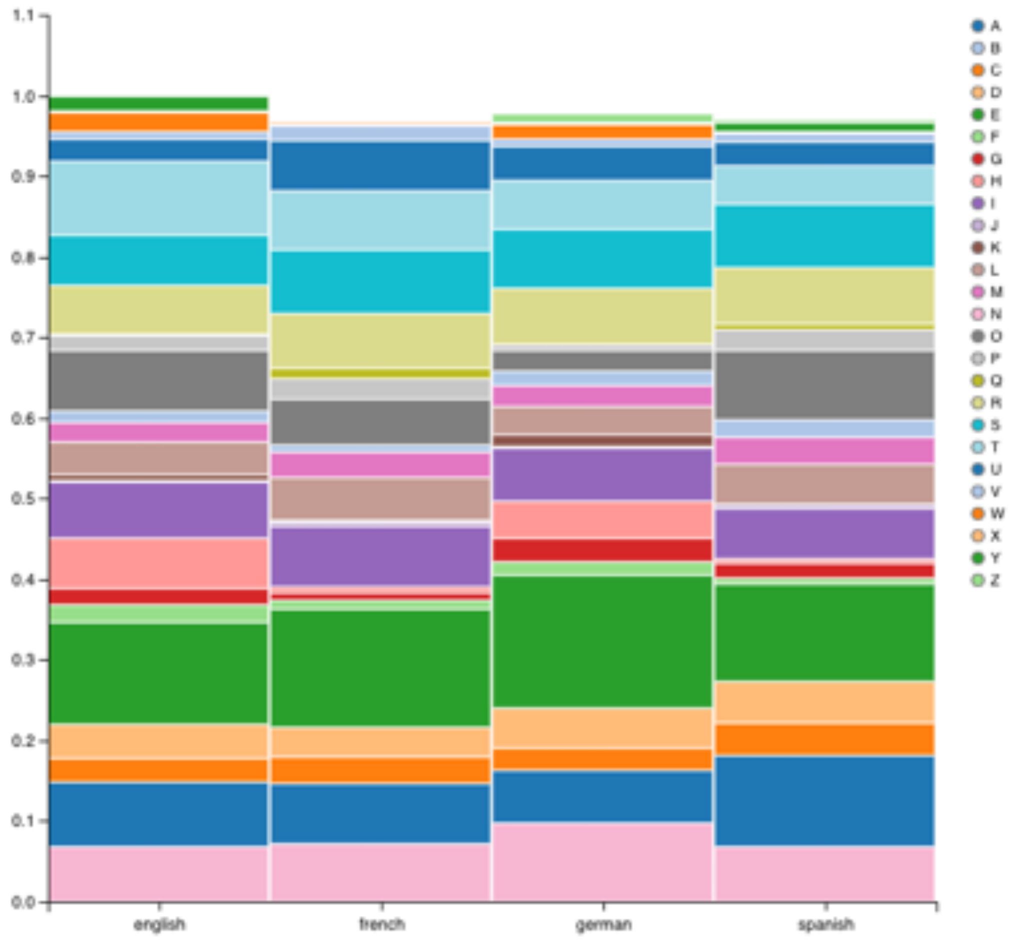
Transforming data onto graphical aesthetics should be easy

Recall: Barchart

```
({:letter "A", :frequency 0.08167, :language "english"}
 {:letter "B", :frequency 0.01492, :language "english"}
 {:letter "C", :frequency 0.02782, :language "english"}
 {:letter "D", :frequency 0.04253, :language "english"}
 {:letter "E", :frequency 0.12702, :language "english"})
```

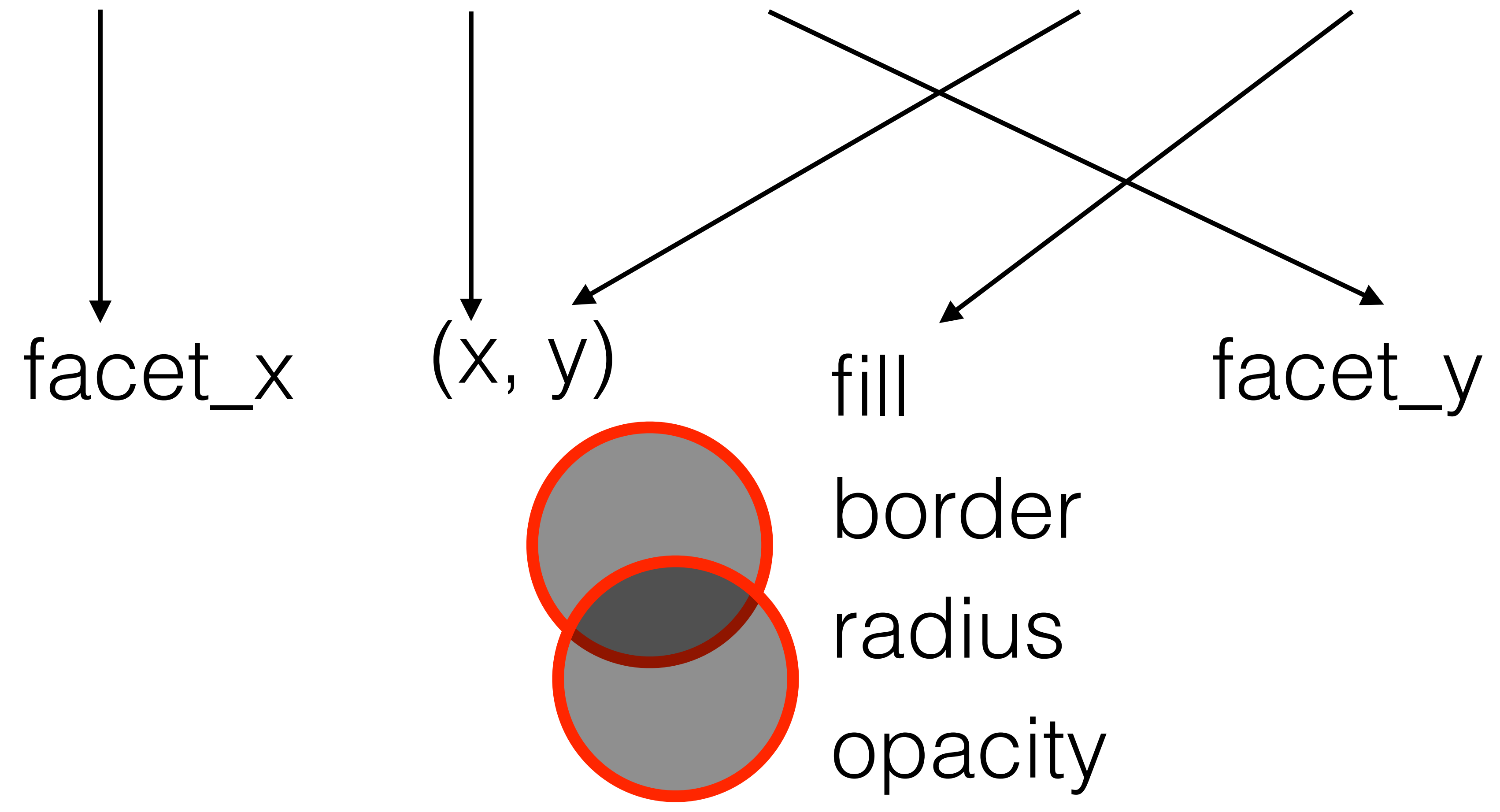


```
(view/plot!
  (gyptis/stacked-bar letter-frequency
                      {:x    :language,
                       :y    :frequency
                       :fill :letter}))
```

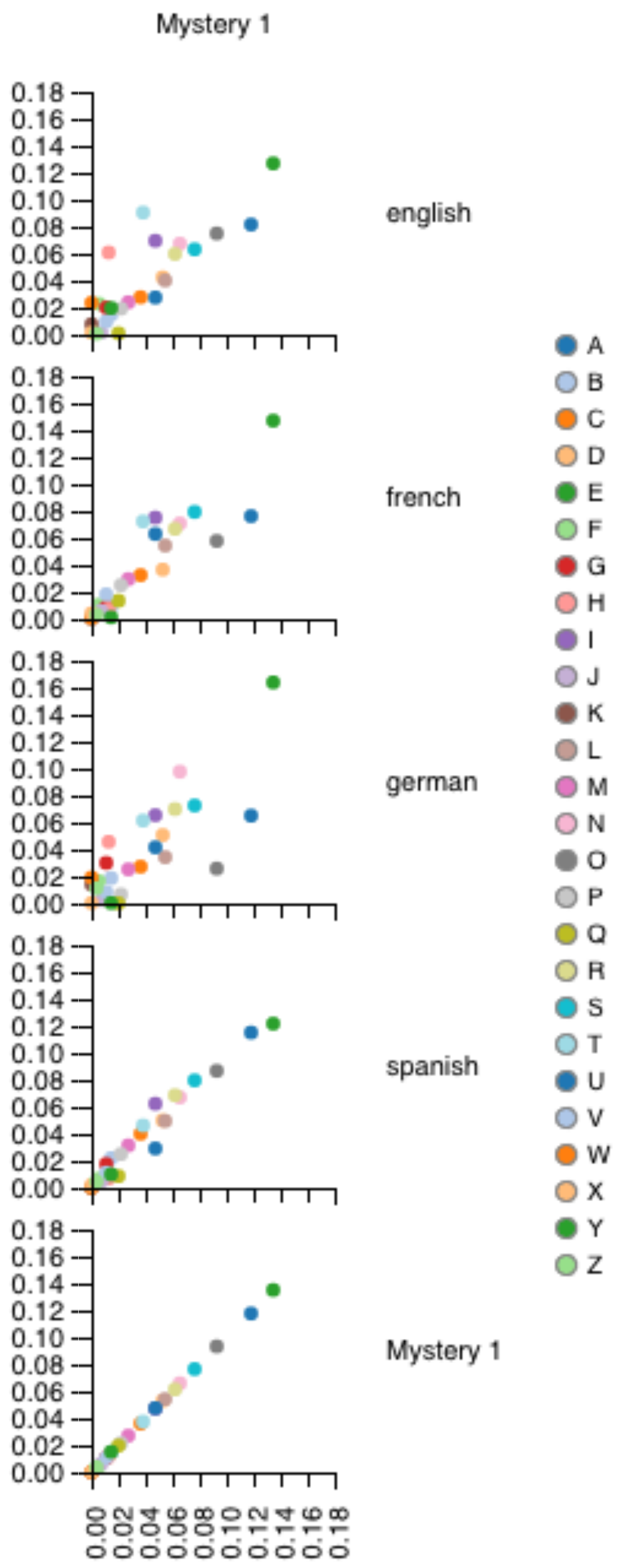


Mystery 1 is in Spanish

```
({:lang_a "Mystery 1", :freq_a 0.00395, :lang_b "Mystery 1", :freq_b 0.00395, :letter "Z"}
 {:lang_a "Mystery 1", :freq_a 0.00395, :lang_b "spanish", :freq_b 0.00467, :letter "Z"}
 {:lang_a "Mystery 1", :freq_a 0.00395, :lang_b "german", :freq_b 0.01134, :letter "Z"})
```



```
(-> letter-cross-product
    (gyptis/point {:x :freq_a :y :freq_b :fill :letter})
    gyptis.vega-templates/vertical-x-labels
    (gyptis/facet-grid {:facet_x :lang_a :facet_y :lang_b})
    (assoc-in [:legends 0 :orient] "left")
    view/plot!
    :legends)
```



Gytpis is alpha!
(suggestions welcome)

Questions?