DYALOG

Elsinore 2023

Vega Charts with Dyalog

Rich Park

Grammar of Graphics

Wilkinson, Leland. *The grammar of graphics*. Springer Berlin Heidelberg, 2012.

Many graphics libraries implement specific and opaque chart types

How would a more formal specification of graphics work?

Grammar of Graphics

Wilkinson, Leland. *The grammar of graphics*. Springer Berlin Heidelberg, 2012.

Map data onto co-ordinates or other values as *graphs*

$$f: A \to B \qquad \{(a, f(a)) : a \in A\} \subseteq A \times B$$

Aesthetic functions translate graphs into *graphics* to create charts

Vega

A visualisation grammar

- Specify data as a table
- Scale functions create the "graph"
- Axes visualise scales using ticks, grid lines...
- Mark properties (x, y, size...) are mapped to scales
- Signals are dynamic variables used for interaction

```
{
  "$schema": "https://vega.github.io/schema/vega/v5.json",
  "width": 400,
  "height": 200,
  "padding": 5,

  "data": [
    {
      "name": "table",
      "values": [
        {"category": "A", "amount": 28},
        {"category": "B", "amount": 55},
        {"category": "C", "amount": 43},
        {"category": "D", "amount": 91},
        {"category": "E", "amount": 81},
        {"category": "F", "amount": 53},
        {"category": "G", "amount": 19},
        {"category": "H", "amount": 87}
      ]
    }
  ],

  "signals": [
```

```
    "encode": {
     "enter": {
      "align": {"value": "center"},
      "baseline": {"value": "bottom"},
      "fill": {"value": "#333"}
     },
     "update": {
      "x": {"scale": "xscale", "signal": "tooltip.category", "band": 0.5},
      "y": {"scale": "yscale", "signal": "tooltip.amount", "offset": -2},
      "text": {"signal": "tooltip.amount"},
      "fillOpacity": [
       {"test": "isNaN(tooltip.amount)", "value": 0},
       {"value": 1}
      ]
     }
    }
   }
  ]
}
```

Dyalog Vega

Specifications are expressed in JSON

```
Vega ← ⎕JSON
```

Dyalog Vega

Any Questions?

Dyalog Vega

Any Questions¿

Vega-Lite

A high-level grammar of interactive graphics

Vega-Lite → Vega → D3 → JavaScript

Vega-Lite

Map data onto graphical marks

- Mark types (point, line, bar, arc...)
- Encoding channels (x, y, color, size...)

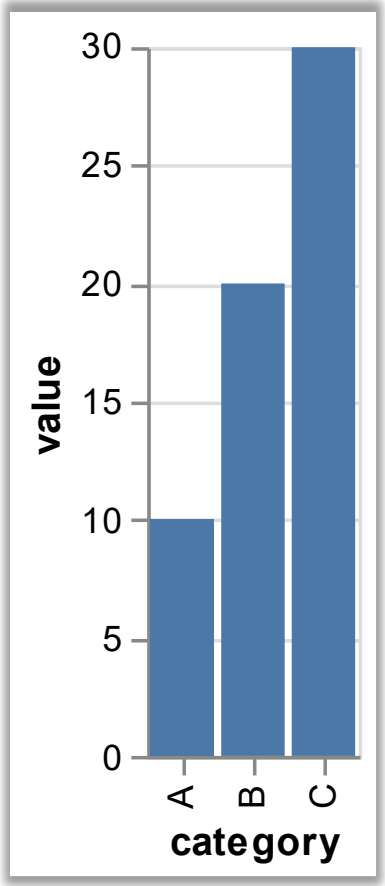
Transform data

- Aggregate, filter, pivot, stack...

Compose charts

- Repeat, Facet, Vconcat, Hconcat...

```
{
  "data": {
    "values": [ {"category": "A", "value": 10},
                {"category": "B", "value": 20},
                {"category": "C", "value": 30}
              ]
  },
  "encoding": {
    "x": {
      "field": "category",  "type": "nominal"
    },
    "y": {
      "field": "value",  "type": "quantitative"
    }
  },
  "mark": "bar",
  "$schema": "https://vega.github.io/schema/vega-lite/v5.json"
}
```



Dyalog Interface to Vega-Lite

- Data from APL tables
- Convert JSON text ← → namespace
- Functions to set parameters
- Pre-made chart functions – just add data!
- HTMLRenderer
  - Preview
  - Render as PNG / SVG

Data

Data

Tables & Data Types

Year	Farm	Variety	Yield
1931	University Farm	Manchuria	27
1931	Waseca	Manchuria	48.86667
1931	Morris	Manchuria	27.43334
1931	Crookston	Manchuria	39.93333
1931	Grand Rapids	Manchuria	32.96667

Data

Tables & Data Types

Year	Farm	Variety	Yield
1931	University Farm	Manchuria	27
1931	Waseca	Manchuria	48.86667
1931	Morris	Manchuria	27.43334
1931	Crookston	Manchuria	39.93333
1931	Grand Rapids	Manchuria	32.96667

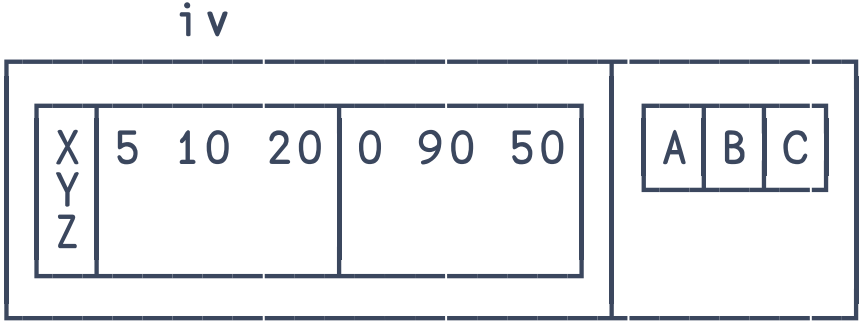
quantitative, nominal, ordinal, temporal, geojson

Data

```
      data
A   B   C
X   5   0
Y  10  90
Z  20  50
```

```
      data ⎕CSV ''
A,B,C
X,5,0
Y,10,90
Z,20,50
```

Data




```
      iv ⎕CSV ''
A,B,C
X,5,0
Y,10,90
Z,20,50
```

Quick Chart functions

Create common chart types from specific data format

Bar

Line

Scatter

Pie

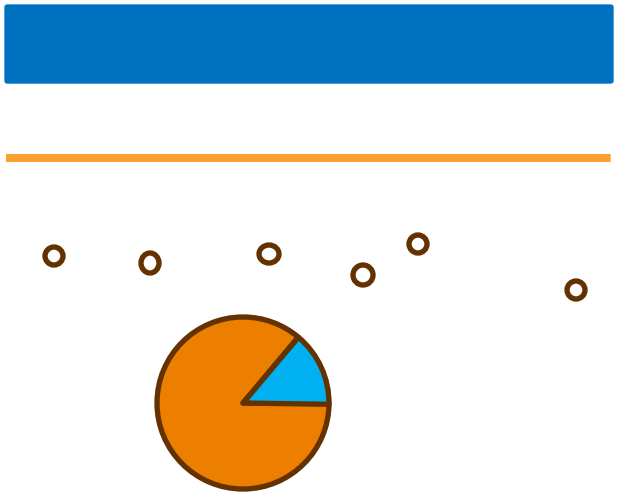


Chart.Bar

Default encoding

Column 1	Column 2	Column 3 (optional)
x	y	color

Chart.Bar

Default encoding

Column 1	Column 2	Column 3 (optional)
x	y	color

```
      □←data←'AB';'XYZ',[0.5]5 10 20
A  B
X  5
Y 10
Z 20
```

Chart.Bar

Default encoding

Column 1	Column 2	Column 3 (optional)
x	y	color

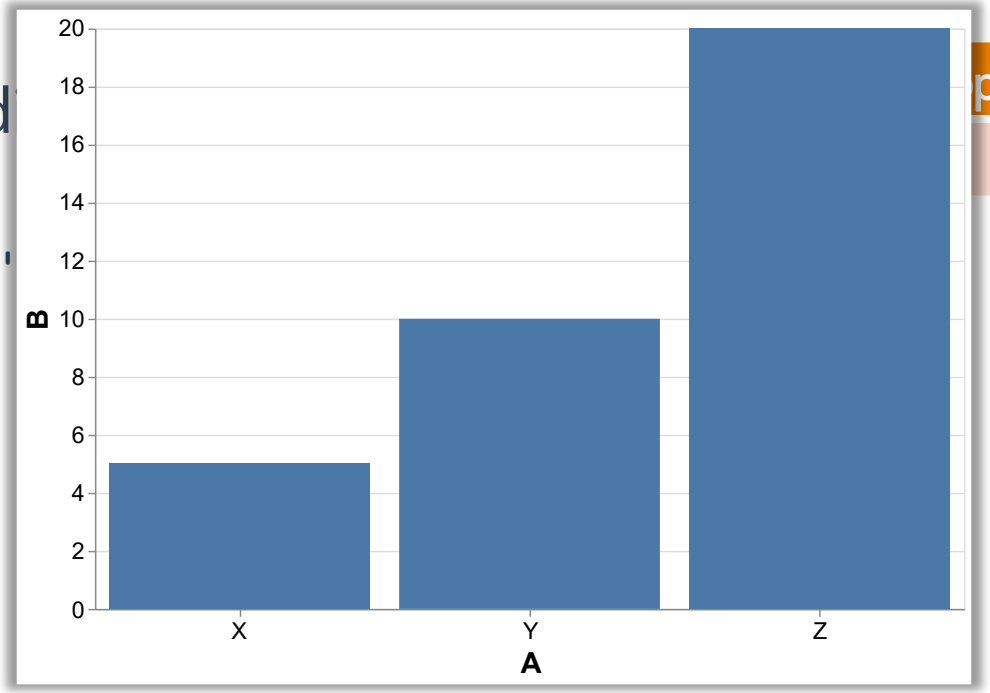
```
      ⍞←data←'AB';'XYZ',[0.5]5 10 20
A  B
X  5
Y 10
Z 20

      View Chart.Bar data
```

Chart.Bar

Default encod

ptional)

```
⍞←data←'
A  B
X  5
Y 10
Z 20
```




```
View Chart.Bar data
```

Chart.Bar

Default encoding

Column 1	Column 2	Column 3 (optional)
x	y	color

```
      data,←'C' 0 90 50

      data

A  B  C
X  5  0
Y 10 90
Z 20 50
       View Chart.Bar data
```

Chart.Bar

Default en

```
      data
      data
A   B   C
X   5   0
Y  10  90
Z  20  50
        Vie
```

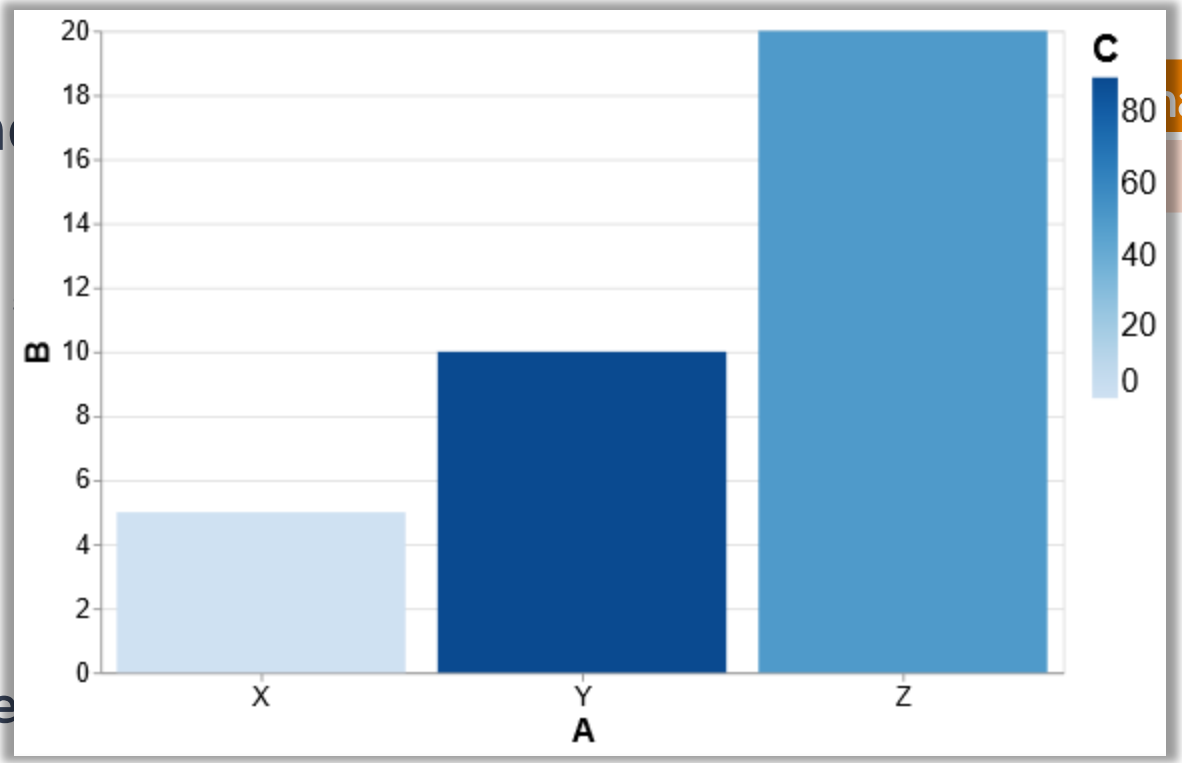


Chart.Bar

```
Bar←{
    ##.Spec.(Encode'bar'Mark Chart)⍵
}
```

Chart.Line

```
      x ← (÷×(○2×⍳)) 1000
      y ← 1○x

View Chart.Line 'x' 'sine'⍪x,⍪y
```

Chart.Line

```
x ←
y ←

View Char
```

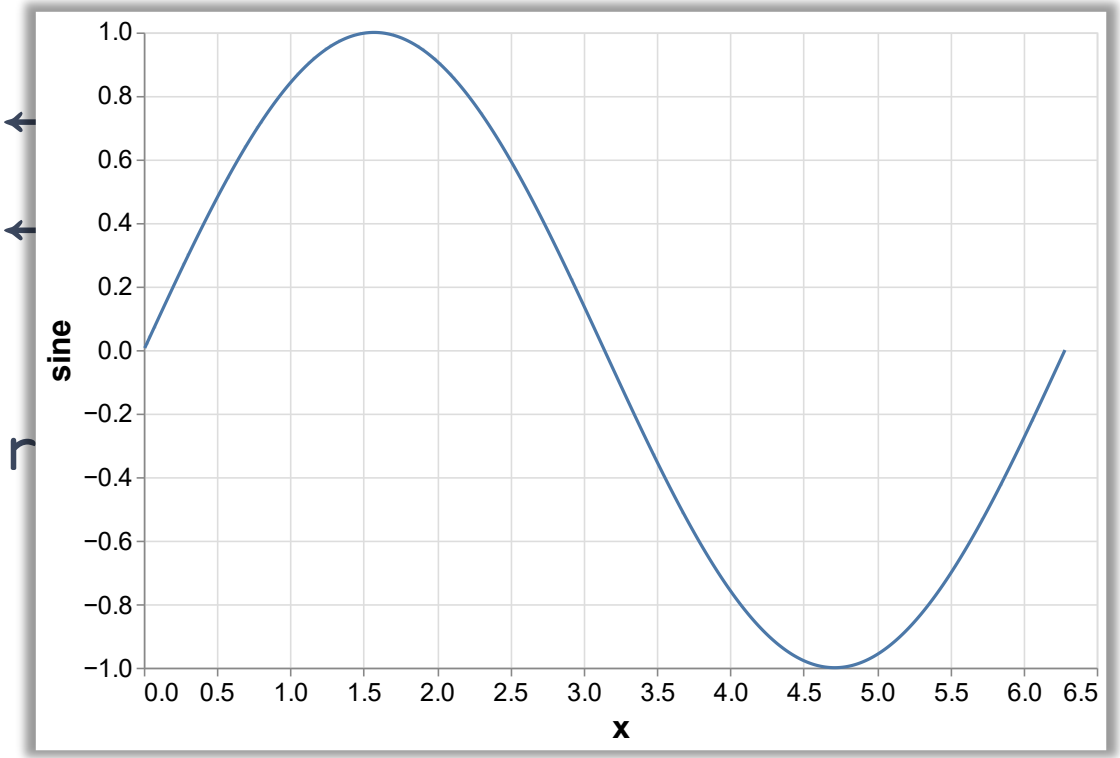


Chart.Line

```
x ← (÷×(○2×⍳)) 1000
y ← ⍉ 1 2 ∘.○ x
data ← 'x' 'sine' 'cos'⍪x,y
```

Chart.Line

x	sine	cos
0.006283185307	0.006283143966	0.9999802609
0.01256637061	0.01256603988	0.9999210442
0.01884955592	0.01884843972	0.9998223524
0.02513274123	0.02513009544	0.9996841893
0.03141592654	0.03141075908	0.9995065604

Chart.Line

```
x ← (÷×(○2×⍳)) 1000
y ← ⊖ 1 2 ∘.○ x
data ← 'x' 'sine' 'cos'⍪x,y
Transform.Stack data
```

Chart.Line

```
x ←
y ←
dat
Tra
```

x	y	function
0.006283185307	0.006283143966	sine
0.01256637061	0.01256603988	sine
0.01884955592	0.01884843972	sine
0.02513274123	0.02513009544	sine
0.03141592654	0.03141075908	sine
⋮	⋮	⋮
6.283185307	1	cos

Chart.Line

```
x ← (÷×(○2×⍳)) 1000
y ← ⍉ 1 2 ∘.○ x
data ← 'x' 'sine' 'cos'⍪x,y
                    Transform.Stack data
```

Chart.Line

```
x ← (÷×(○2×⍳)) 1000
y ← ⊖ 1 2 ∘.○ x
data ← 'x' 'sine' 'cos'⍪x,y
View Chart.Line Transform.Stack data
```

Chart.Line

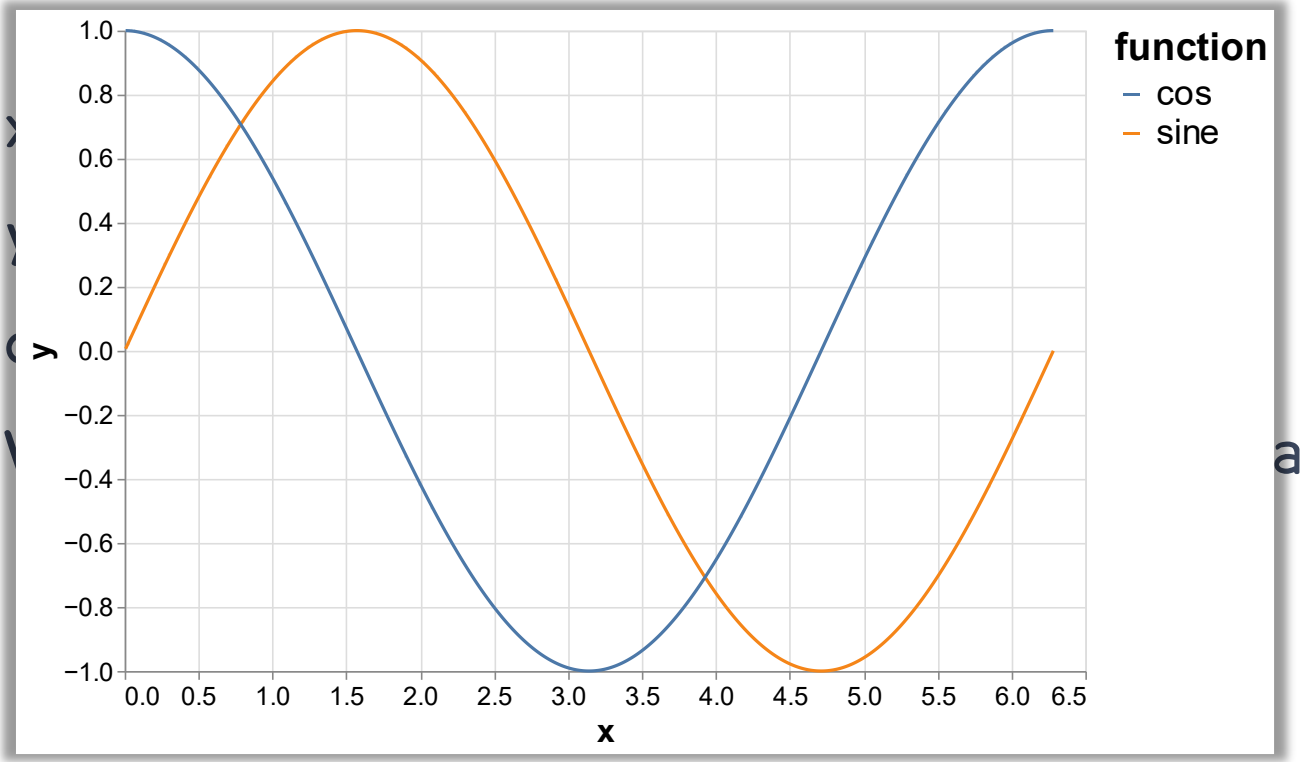

function
cos
sine
y
x
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5

Chart.Line

```
Line←{
      ##.Spec.(Encode'line'Mark Chart)⍵
}
```

Chart.Scatter

```
Scatter←{
    ##.Spec.(Encode'point'Mark Chart)⍵
}
```

Chart.Scatter

```
Scatter
      ##.
}
```

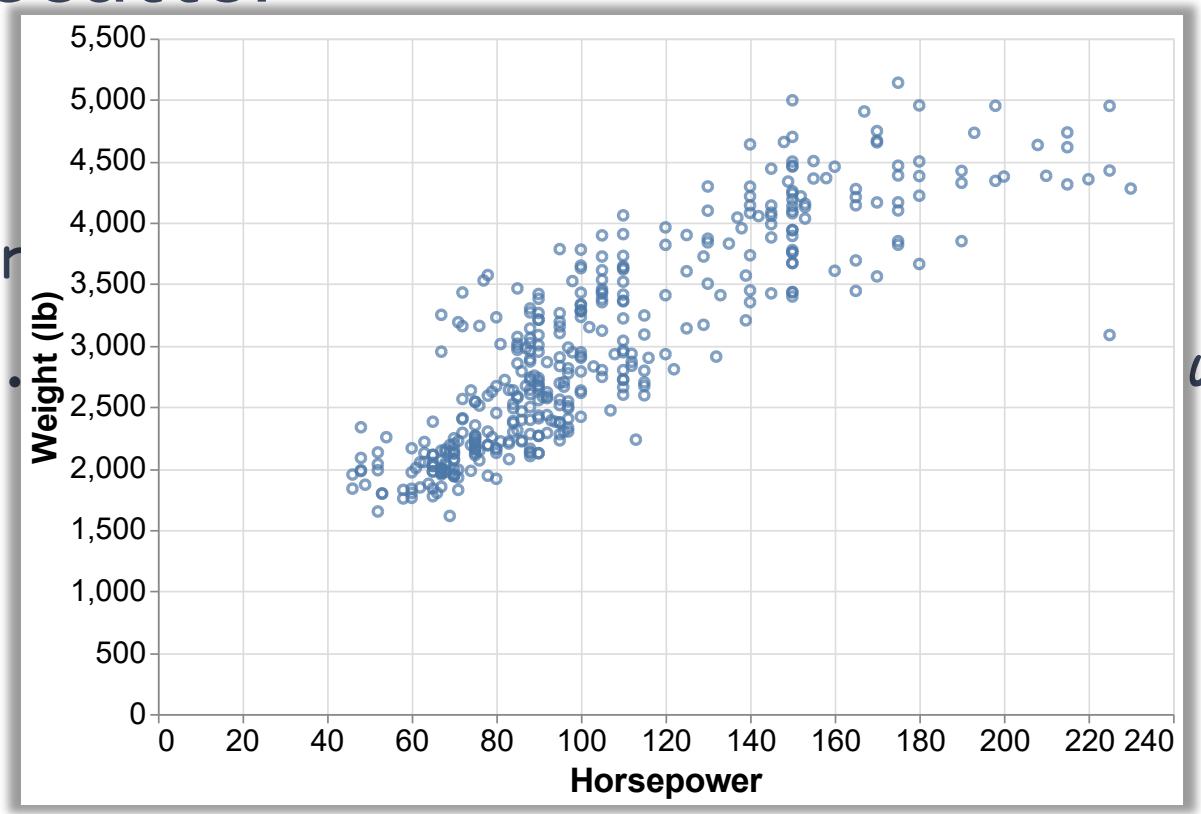


Chart.Pie

```
Pie←{
    ##.Spec.(Encode'arc'Mark Chart)⍵
}
```

Chart.Pie

```
total ← barley[;2],∘(+⌿)⌸barley[;4]
```

Year	Farm	Variety	Yield
1931	University Farm	Manchuria	27
1931	Waseca	Manchuria	48.86667
1931	Morris	Manchuria	27.43334
1931	Crookston	Manchuria	39.93333
1931	Grand Rapids	Manchuria	32.96667

Chart.Pie

Farm	Yield
University Farm	653.33335
Waseca	962.16663
Morris	708.00001
Crookston	748.39997
Grand Rapids	498.63334
Duluth	559.93334

```
total ← barley[;2],∘(+⌿)⌸barley[;4]
```

Chart.Pie

Farm	Yield
University Farm	653.33335
Waseca	962.16663
Morris	708.00001
Crookston	748.39997
Grand Rapids	498.63334
Duluth	559.93334

```
total ← barley[;2],∘(+⌿)⌸barley[;4]
View Chart.Pie total
```

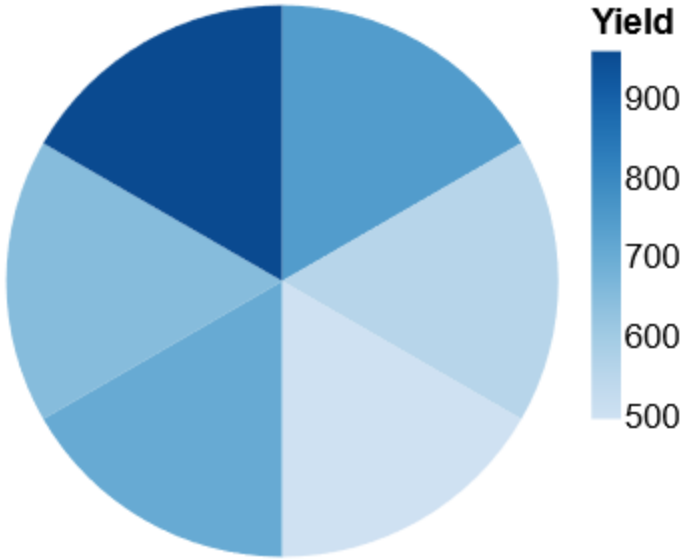
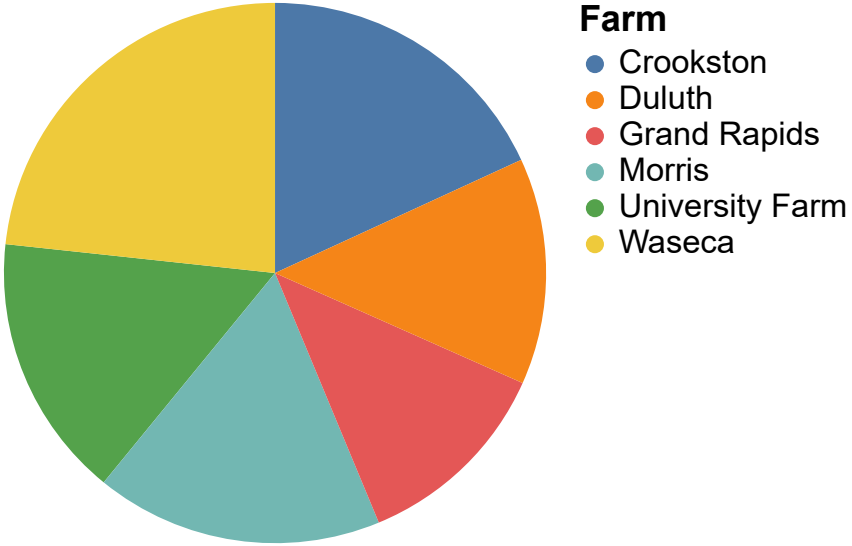


Chart.Pie

Farm	Yield
University Farm	653.33335
Waseca	962.16663
Morris	708.00001
Crookston	748.39997
Grand Rapids	498.63334
Duluth	559.93334

```
total ← barley[;2],∘(+⌿)⌸barley[;4]
View Chart.Pie ⍬total
```



Modified Bar Chart

```
total ← barley[;2 1]{α[1],(+/ω),α[2]}⌸barley[;4]
```

Farm	Yield	Year
University Farm	358.26666	1931
Waseca	543.46666	1931
Morris	292.86669	1931
⋮	⋮	⋮
Duluth	257.00001	1932

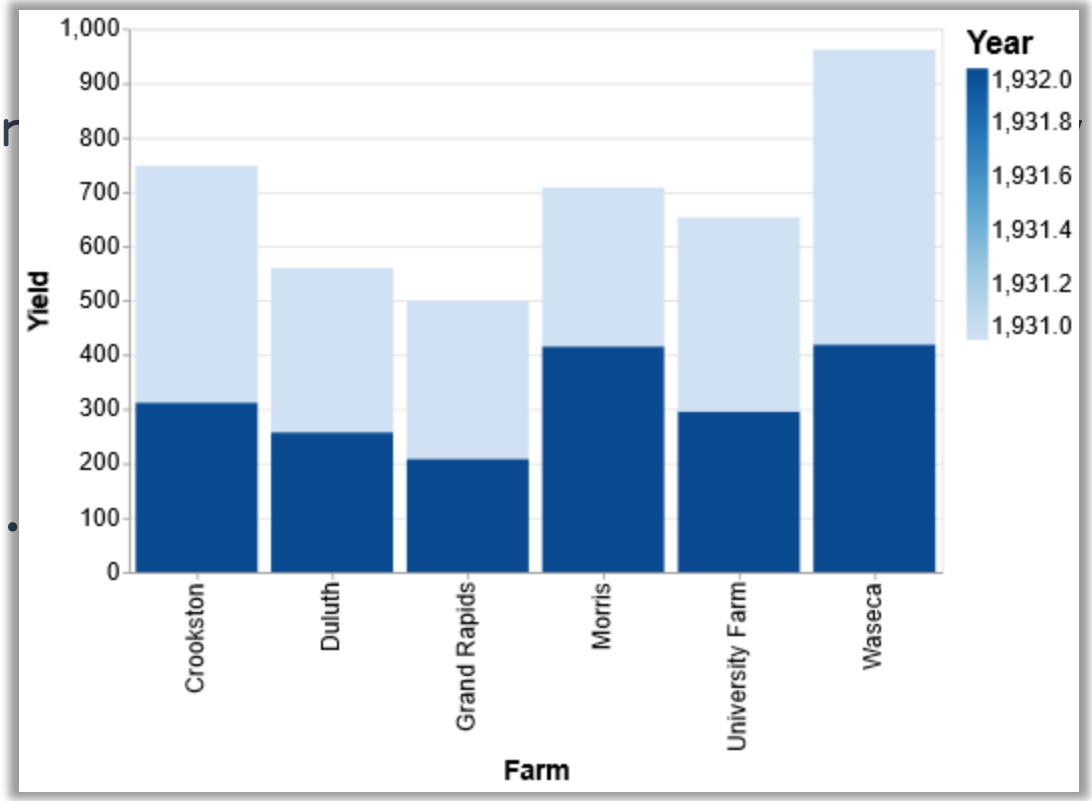
Modified Bar Chart

```
total ← barley[;2 1]{α[1],(+/ω),α[2]}⌸barley[;4]
```

```
View Chart.Bar total
```

Modified Bar Chart

```
total ← bar                                    [;4]

View Chart.
```



Modified Bar Chart

```
total ← barley[;2 1]{α[1],(+/ω),α[2]}⌸barley[;4]


enc←'{xOffset:{field:"Year",type:"nominal"},'
enc,← 'color:{type:"nominal"}}'
```

Modified Bar Chart

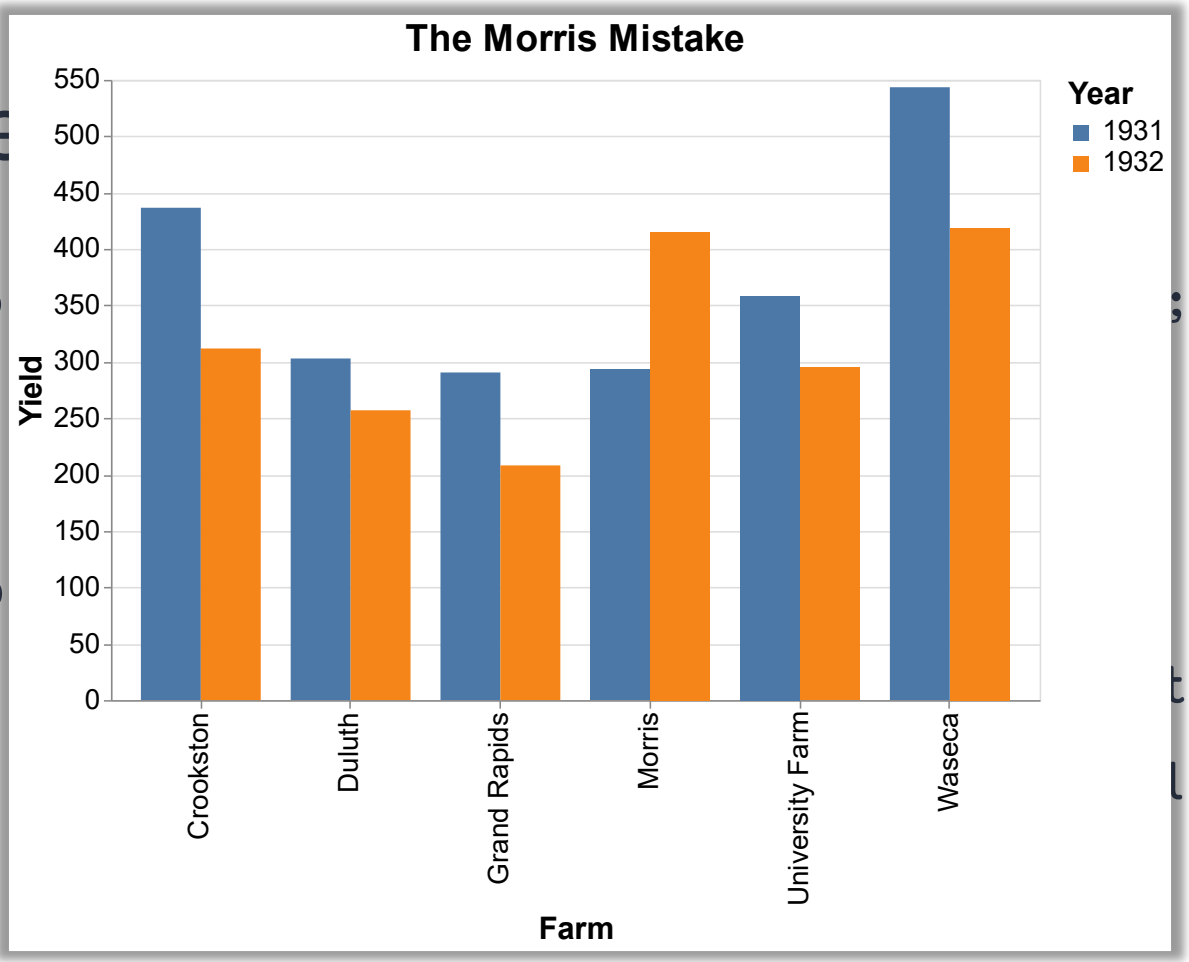
```
total ← barley[;2 1]{⍺[1],(+/⍵),⍺[2]}⌸barley[;4]


enc←'{xOffset:{field:"Year",type:"nominal"},'
enc,← 'color:{type:"nominal"}}'
top_level ← '{title:"The Morris Mistake", fontSize:24}'
```

Modified Bar Chart

```
total ← barley[;2 1]{⍺[1],(+/⍵),⍺[2]}⌸barley[;4]


enc←'{xOffset:{field:"Year",type:"nominal"},'
enc,← 'color:{type:"nominal"}}'
top_level ← '{title:"The Morris Mistake", fontSize:24}'
View enc Spec.Encode top_level Chart.Bar total
```

Modifie

```
total ← b                                    ;4]


enc←'{xOf
enc,← 'co
top_level                                    tSize:24}'
View enc                                     l
```



What to do with this?

- github.com/JoshDavid/VegaLite
- github.com/the-carlisle-group/Playfair

  toolofthought.com/posts/charting-and-tidy-data
- Experiment and refine API
- More quick chart functions
- View compositions
- Interactivity
- An APL Visualisation Grammar?