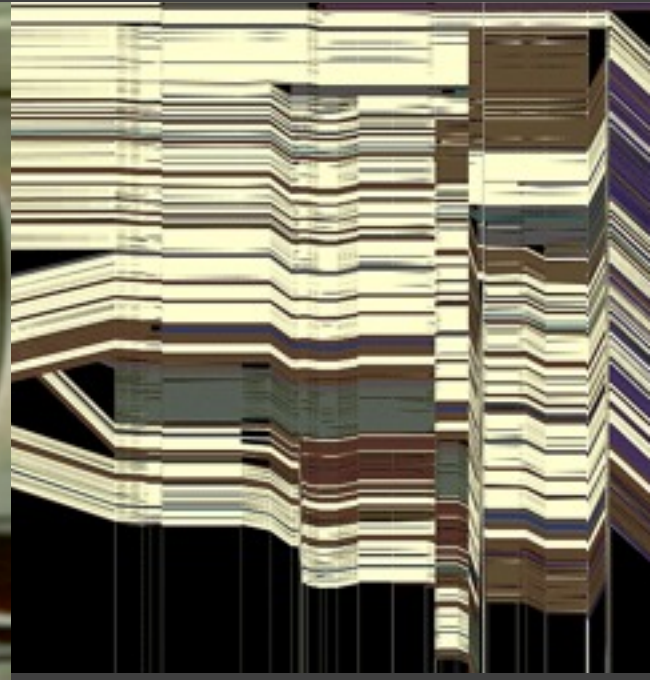
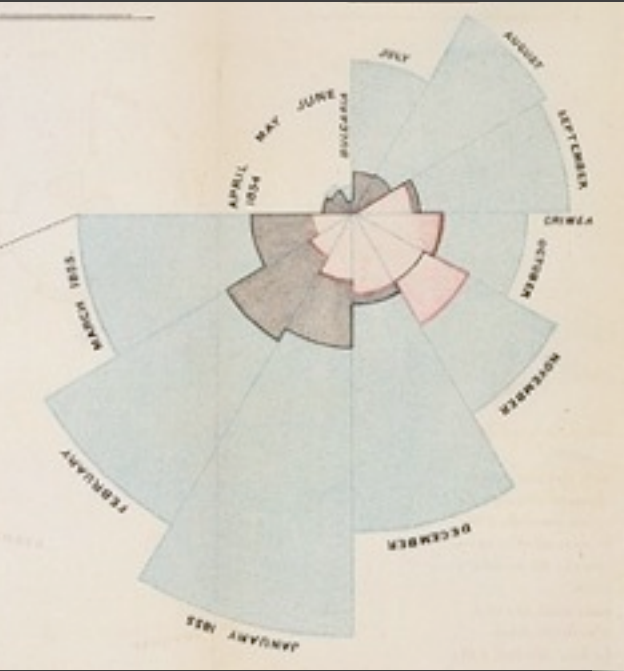


MTI193102 :: 26 June 2021

Visualization Tools



Joko TriLoka

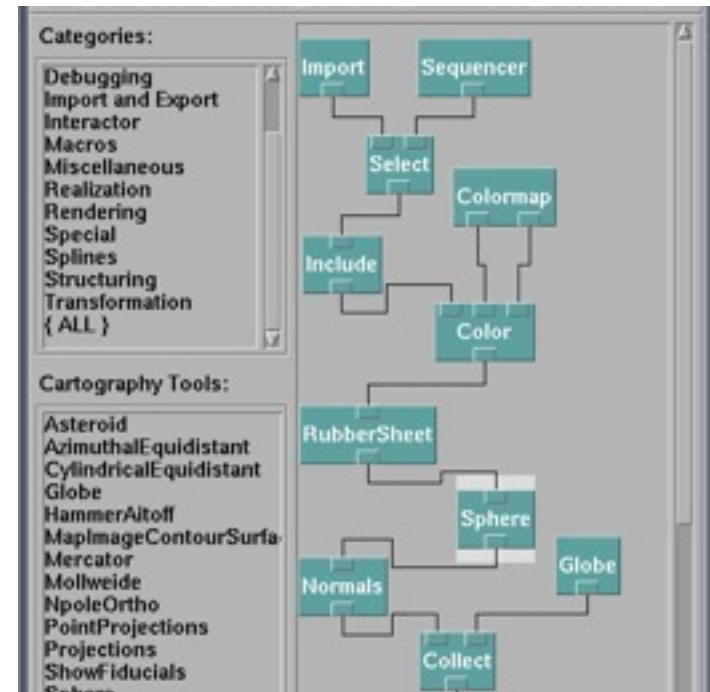
IIB Darmajaya

How do people create visualizations?



Chart Typology

Pick from a stock of templates
Easy-to-use but limited expressiveness
Prohibits novel designs, new data types

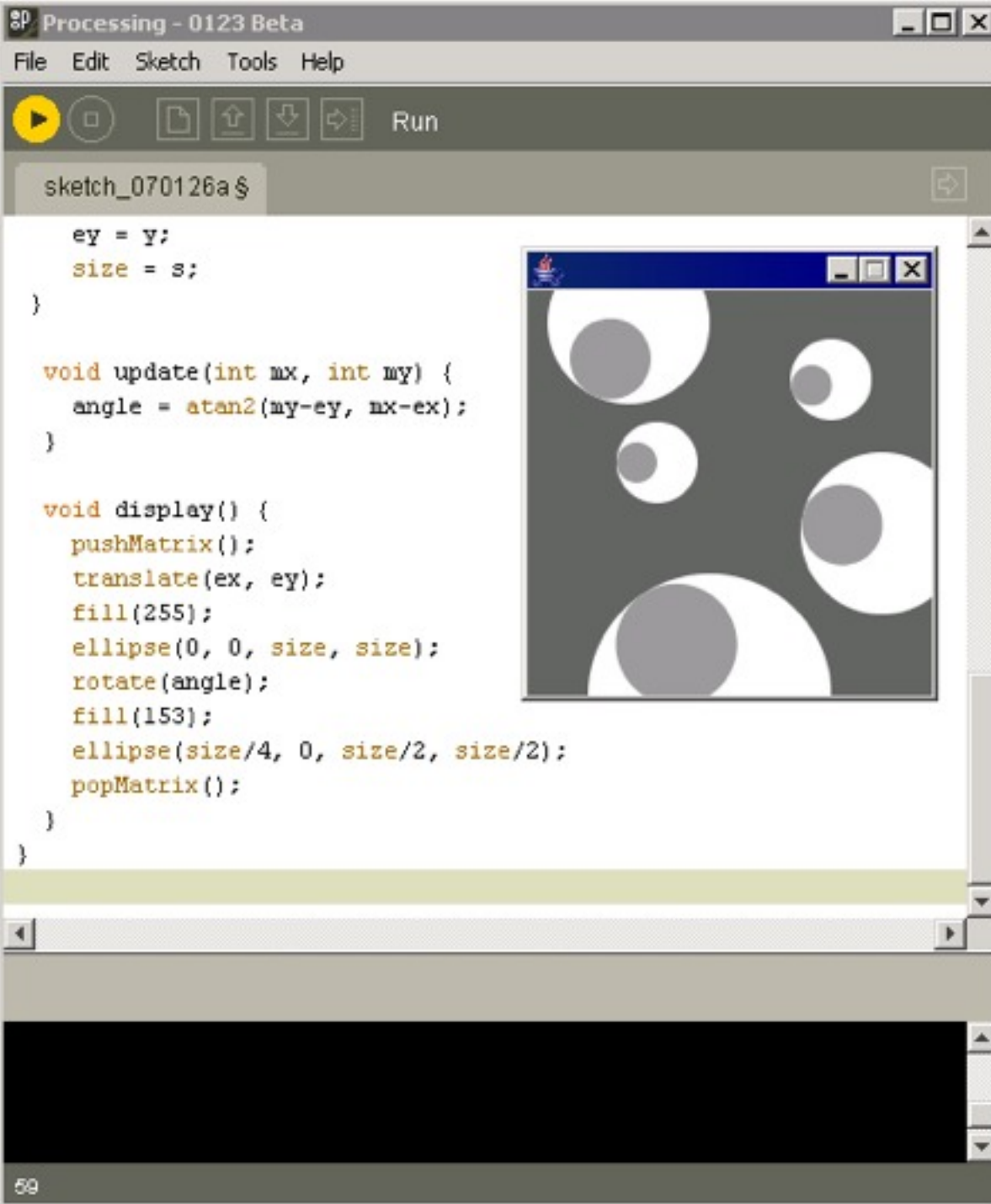


Component Architecture

Permits more combinatorial possibilities
Novel views require new operators,
which requires software engineering.

Graphics APIs

OpenGL, Java2D, GDI+, Processing



<http://processing.org>



US Air Traffic, Aaron Koblin

Graphics APIs

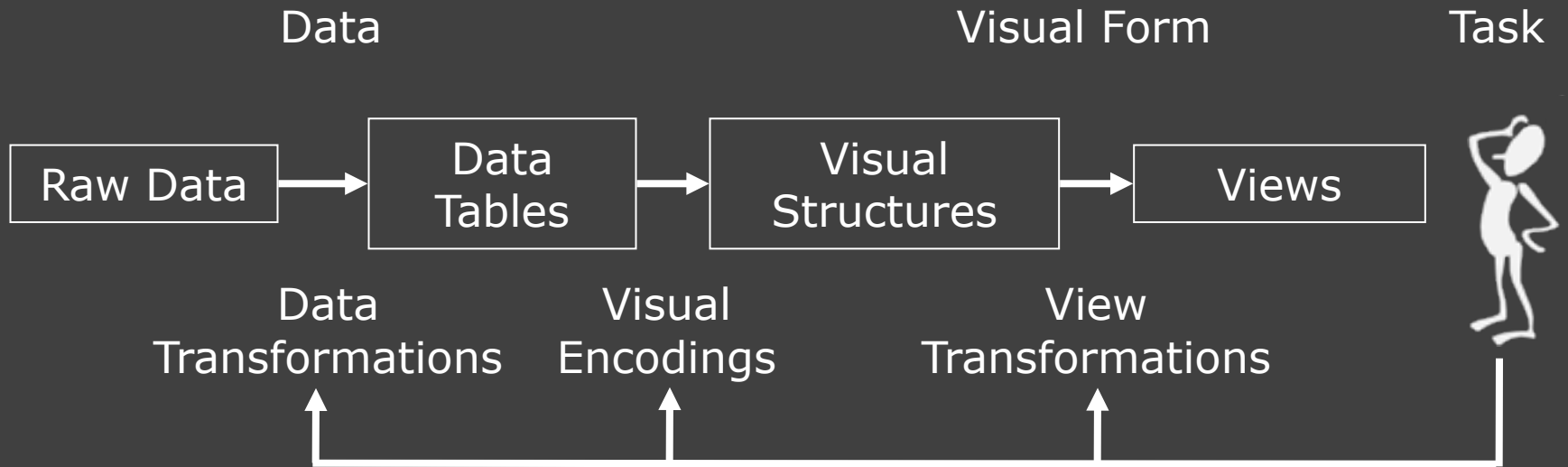
OpenGL, Java2D, GDI+, Processing

Component Model Architectures

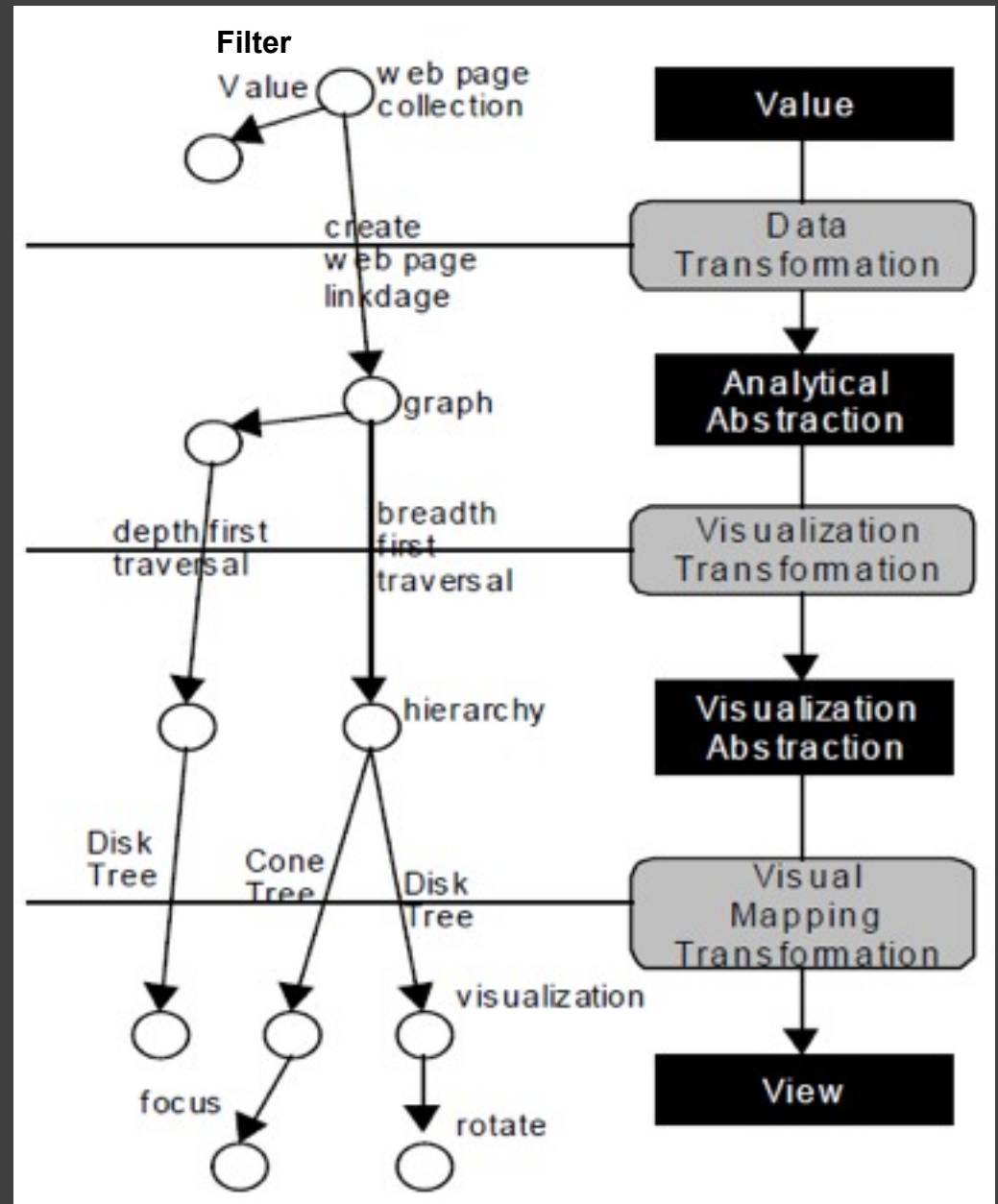
Prefuse, Flare, Improvise

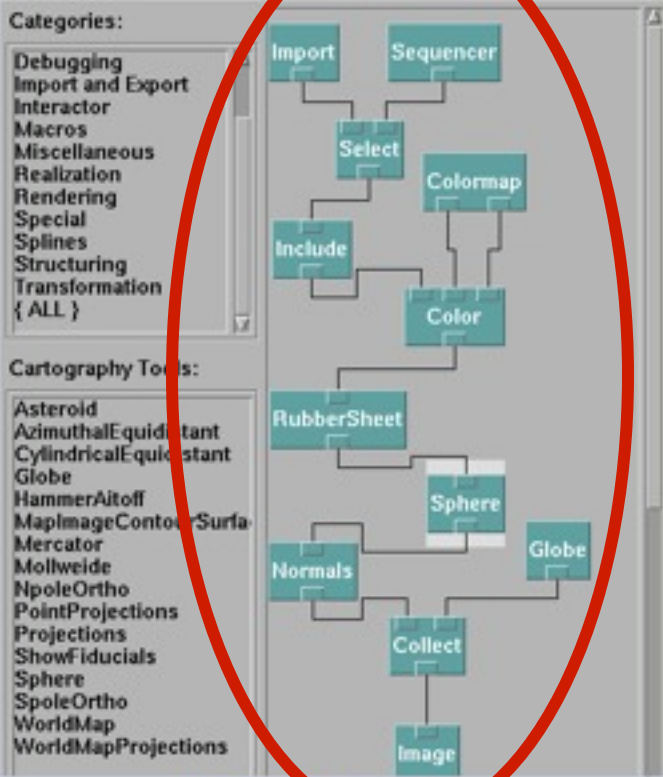
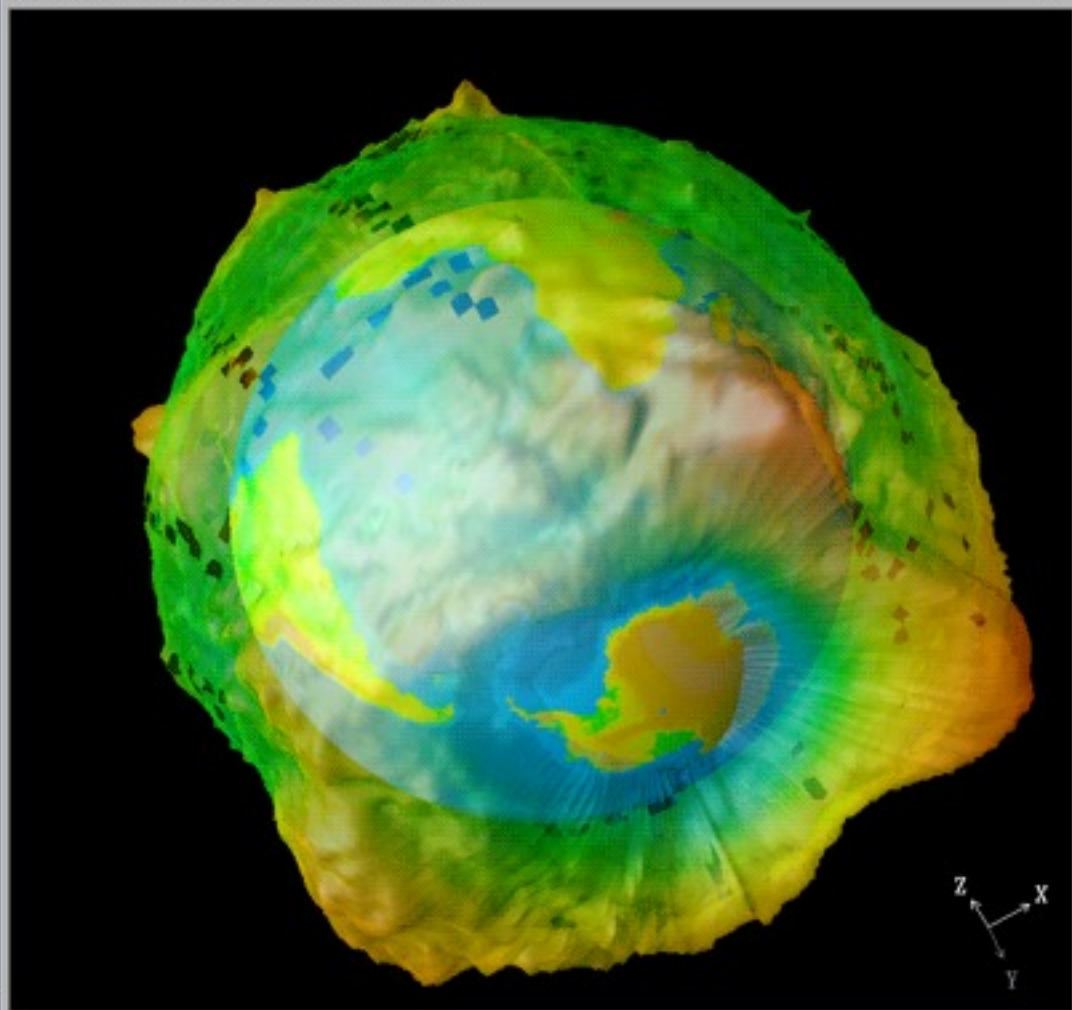
Graphics APIs

OpenGL, Java2D, GDI+, Processing

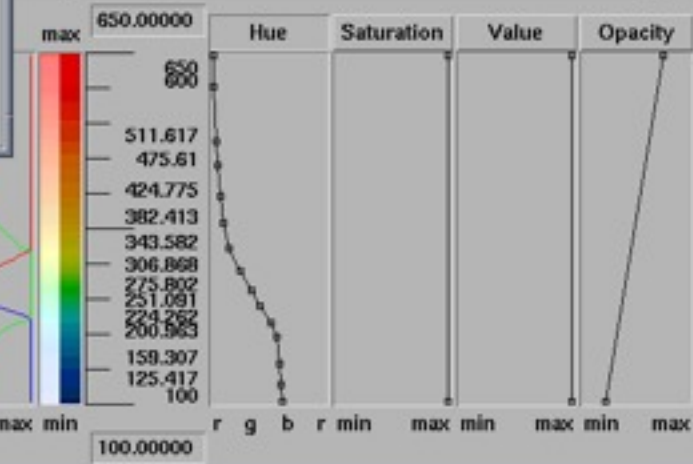


Data State Model [Chi 98]





Colormap Editor



View Control...

Undo Ctrl+U Redo Ctrl+D

Mode: Rotate

Set View: None

Projection: Perspective

View Angle: 30.000

Close Reset Ctrl+F

Sequence Control

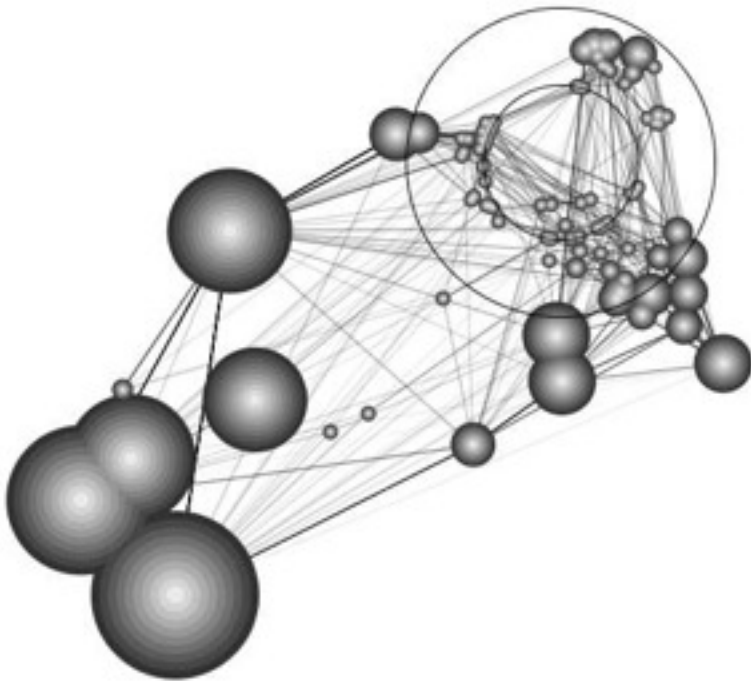
Sequence Control

⏪ ⏩ ⏴ ⏵ ...

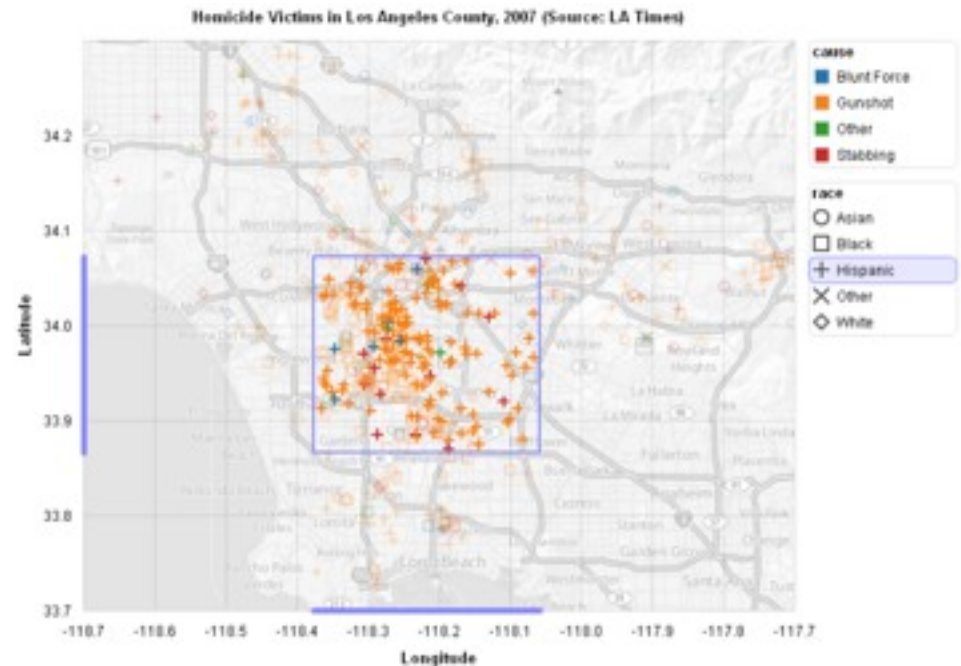
⏮ ⏭ ■ ⏸

Prefuse & Flare

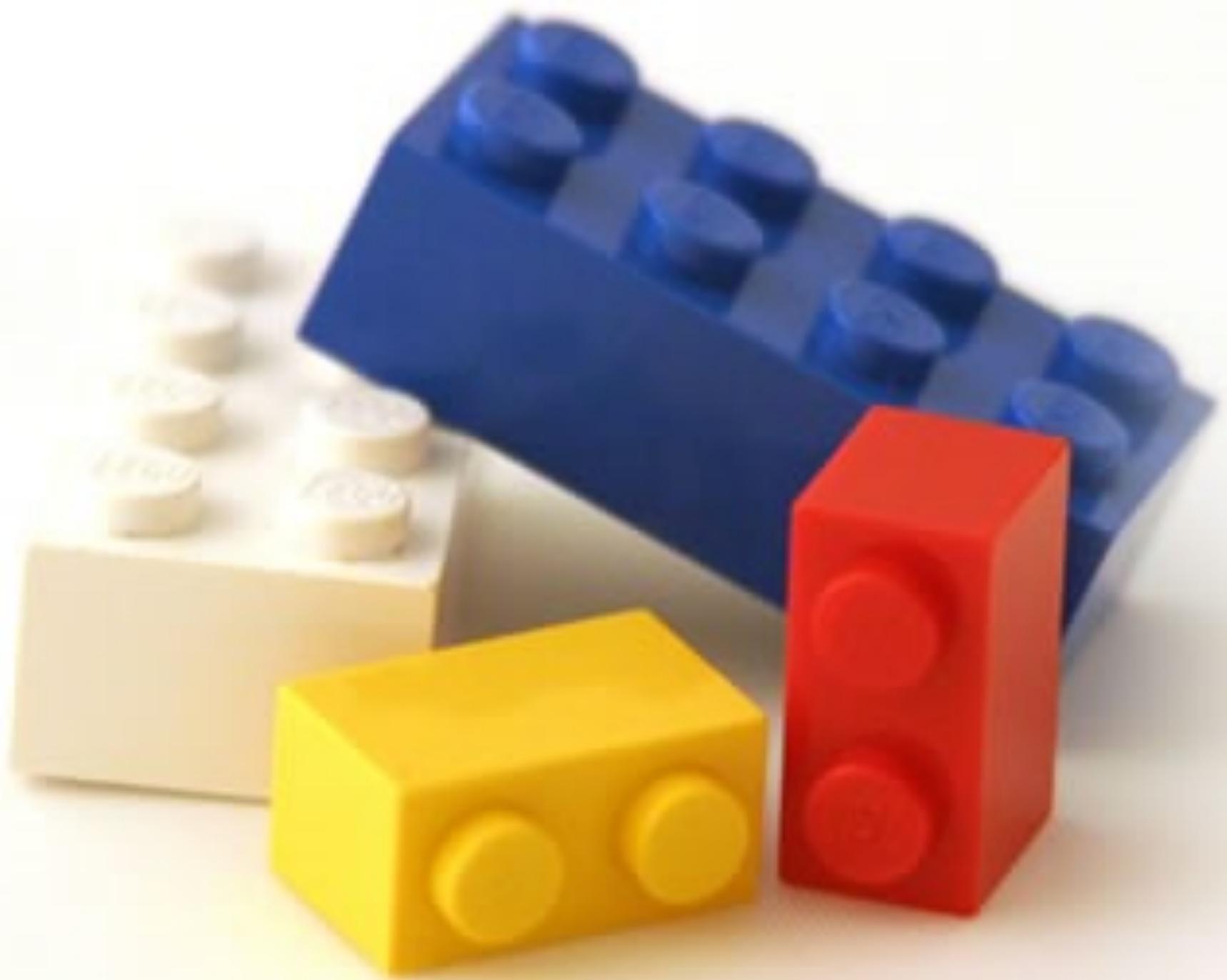
Operator-based toolkits for visualization design
Vis = (Input Data \rightarrow Visual Objects) + Operators



Prefuse (<http://prefuse.org>)



Flare (<http://flare.prefuse.org>)





?

Component Model Architectures

Prefuse, Flare, Improvise

Graphics APIs

OpenGL, Java2D, GDI+, Processing

Chart Typologies

Excel, Many Eyes, Google Charts

Component Model Architectures

Prefuse, Flare, Improvise

Graphics APIs

OpenGL, Java2D, GDI+, Processing



Chart Typologies

Data Sets : State Quick Facts

Uploaded By: [zinggoat](#)

Created at: Friday May 18, 3:08 PM

Data Source: [US Census Bureau](#)

Description:

Tags: [people census](#)

[view as text](#)

[edit data set](#)

	People QuickFacts	Population 2005 estimate	Population percent change April 1 2000 to July 1 2005	Population 2000	Population percent change 1990 to 2000	Persons under 5 years old percent 2004	Persons under 18 years old percent 2004	Persons 65 years old and over percent 2004
1	Alabama	4557808	0.03	4447100	0.1	0.07	0.24	0.13
2	Alaska	663661	0.06	626932	0.14	0.08	0.29	0.06
3	Arizona	5939292	0.16	5130632	0.4	0.08	0.27	0.13
4	Arkansas	2779154	0.04	2673400	0.14	0.07	0.25	0.14
5	California	36132147	0.07	33871648	0.14	0.07	0.27	0.11
6	Colorado	4665177	0.08	4301261	0.31	0.07	0.26	0.1
7	Connecticut	3510297	0.03	3405565	0.04	0.06	0.24	0.14
8	Delaware	843524	0.08	783600	0.18	0.07	0.23	0.13
9	Florida	17789864	0.11	15982378	0.24	0.06	0.23	0.17
10	Georgia	9072576	0.11	8186453	0.26	0.08	0.26	0.1
11	Hawaii	1275194	0.05	1211537	0.09	0.07	0.24	0.14
12	Idaho	1429096	0.1	1293953	0.29	0.07	0.27	0.11
13	Illinois	12763371	0.03	12419293	0.09	0.07	0.26	0.12



Choosing a visualization type for **State Quick Facts**

Analyze a text



Tag Cloud

How are you using your words? This enhanced tag cloud will show you the words popularity in the given set of text.

[Learn more](#)



Wordle

Wordle is a toy for generating "word clouds" from text that you provide. The clouds give greater prominence to words that appear more frequently in the source text.

[Learn more](#)

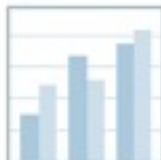


Word Tree

See a branching view of how a word or phrase is used in a text. Navigate the text by zooming and clicking.

[Learn more](#)

Compare a set of values



Bar Chart

How do the items in your data set stack up? A bar chart is a simple and recognizable way to compare values. You can display several sets of bars for multivariate comparisons.

[Learn more](#)



Block Histogram

This versatile chart lets you get a quick sense of how a single set of data is distributed. Each item in the data is an individually identifiable block.

[Learn more](#)

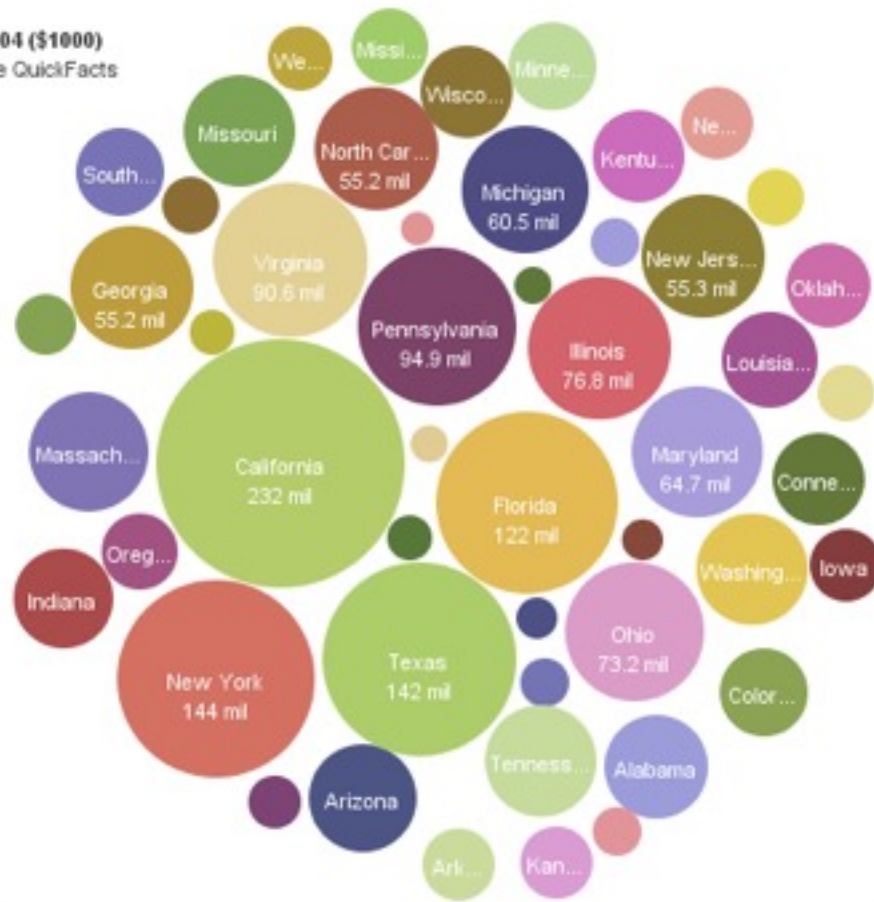
Visualizations : Federal Spending by State, 2004

Creator: Anonymous
 Tags: **census people**

People QuickFac...
 Click to select,
 Ctrl-Click: multiple
 Shift-Click: range

Federal spending 2004 (\$1000)
 Disks colored by People QuickFacts

- Alabama
- Alaska
- Arizona
- Arkansas
- California
- Colorado
- Connecticut
- Delaware
- Florida
- Georgia
- Hawaii
- Idaho
- Illinois
- Indiana
- Iowa
- Kansas
- Kentucky
- Louisiana
- Maine
- Maryland



Search>>

To highlight or find totals
 click or ctrl-click.

Bubble Size: **Federal spending 2004 (\$1000)** | Label: **People QuickFacts** | Color: **People QuickFacts**

Data file: **Census Bureau** This data set has not yet been rated

full image rate this

- Retail sales per capita 2002
- Minority-owned firms percent of total 1997
- Women-owned firms percent of total 1997
- Housing units authorized by building permits 2004
- Federal spending 2004 (\$1000)**
- Land area 2000 (square miles)
- Persons per square mile 2000
- FIPS Code



MAD LIBS®

MY MUSIC LESSON

Every Wednesday, when I get home from school, I have a piano lesson. My teacher is a very strict house. Her name is

Hillary Clinton. Our piano is a Steinway Concert tree

and it has 88 ~~keys~~ cups. It also has a soft pedal and a/an

Smily pedal. When I have a lesson, I sit down on the piano

AIBERTO and play for 16 minutes. I do scales to

exercise my cats, and then I usually play a minuet by

Johann Sebastian washington. Teacher says I am a natural

Haunted House and have a good musical leg. Perhaps

when I get better I will become a concert vet and give

a recital at Carnegie hospital.

[M]ost charting packages channel user requests into a rigid array of chart types. To atone for this lack of flexibility, they offer a kit of post-creation editing tools to return the image to what the user originally envisioned. They give the user an impression of having explored data rather than the experience.

Leland Wilkinson, *The Grammar of Graphics*

Chart Typologies

Excel, Many Eyes, Google Charts

Component Model Architectures

Prefuse, Flare, Improvise

Graphics APIs

OpenGL, Java2D, GDI+, Processing

Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Languages

Tableau VizQL, ggplot2

Component Model Architectures

Prefuse, Flare, Improvise

Graphics APIs

OpenGL, Java2D, GDI+, Processing

Schema: congress.csv Connection

Find:

Dimensions

- Abc Candidate
- Abc Candidate ID
- Abc General Elec Status
- Abc Incumbent/Challenger/Open-Seal
- # Party
- Abc Party Desig
- Abc Primary Elec Status
- Abc Runoff Elec Status
- Abc Spec Elec Status
- Abc State Code
- # Year
- Abc Measure Names

Measures

- # District
- # General Elec Pct
- # Total Receipts
- # Measure Values

Groups

Filters:

Level of Detail:

Mark: Automatic

Text:

Color: Party

Size:

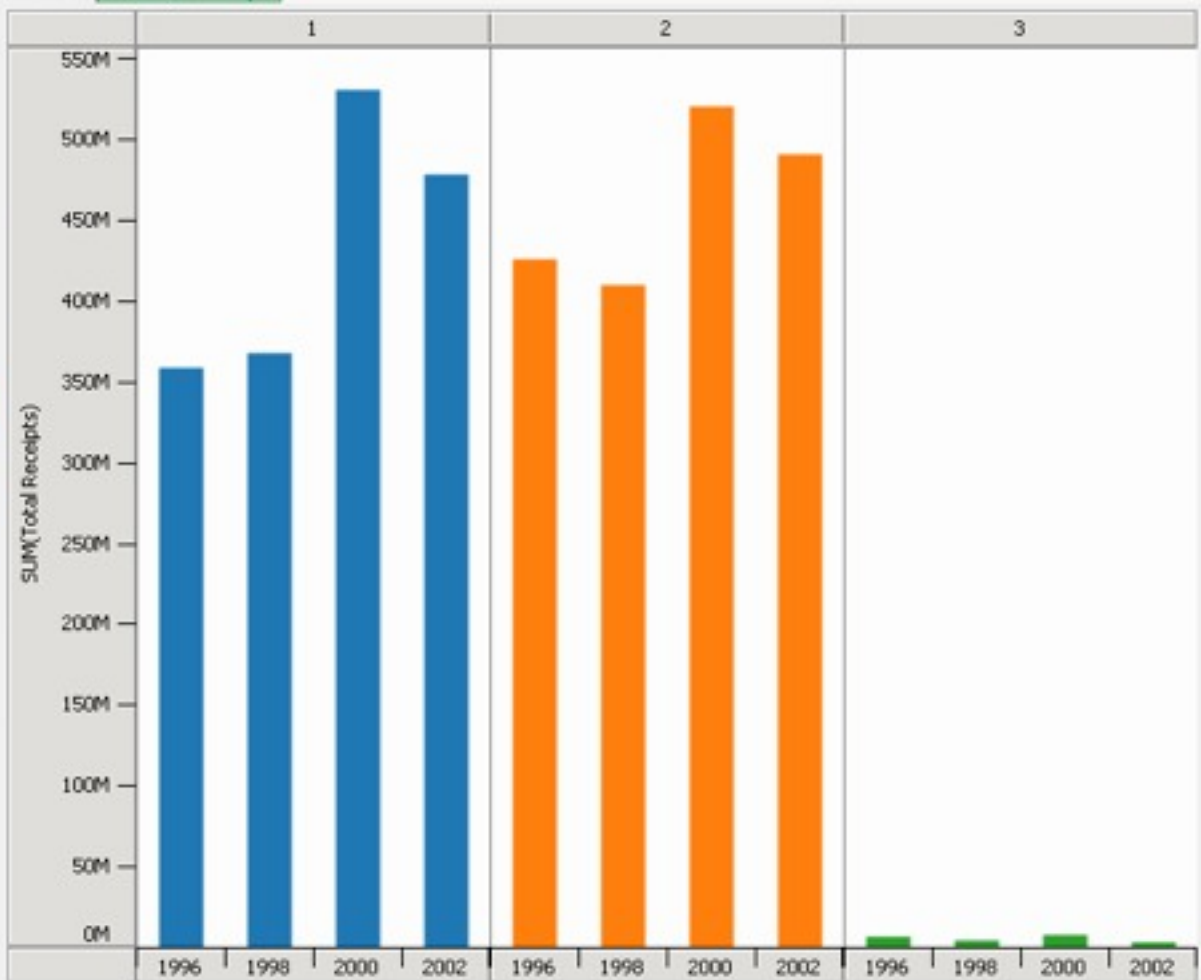
Legend:

- 1
- 2
- 3

Size:

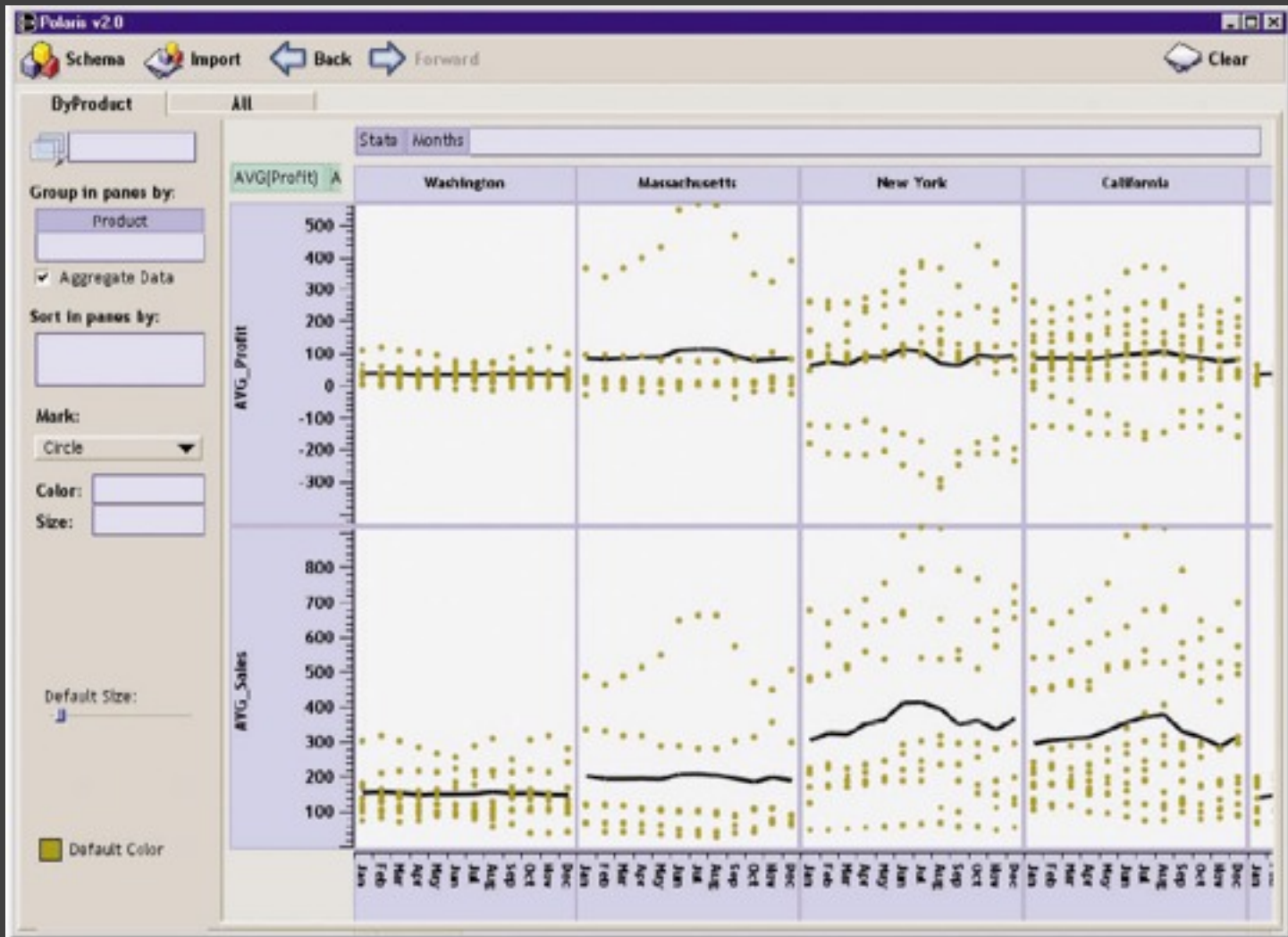
Columns: Party Year

Rows: SUM(Total..)



Polaris

Research at Stanford by Stolte, Tang, and Hanrahan.



Polaris / Tableau Approach

Insight: can simultaneously specify both database queries and visualization

Choose data, then visualization, not vice versa

Use smart defaults for visual encodings

More recently: automate visualization design

Specifying Table Configurations

Operands are the database fields

- Each operand interpreted as a set {...}
- Quantitative and Ordinal fields treated differently

Three operators:

- **concatenation (+)**
- **cross product (x)**
- **nest (/)**

Table Algebra: Operands

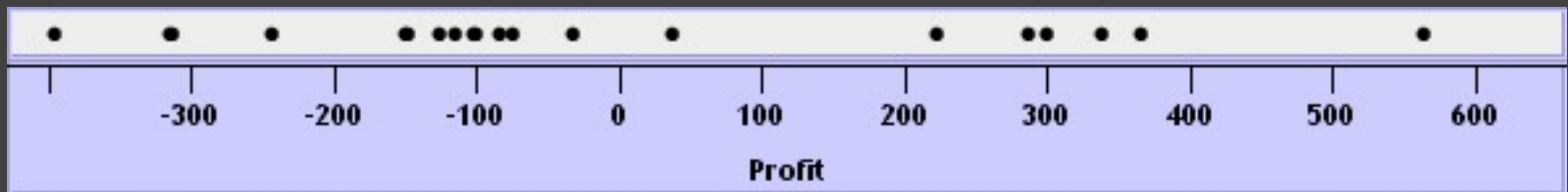
Ordinal fields: interpret domain as a set that partitions table into rows and columns.

Quarter = {(Qtr1),(Qtr2),(Qtr3),(Qtr4)} →

Qtr1	Qtr2	Qtr3	Qtr4
95892	101760	105282	98225

Quantitative fields: treat domain as single element set and encode spatially as axes:

Profit = {(Profit[-410,650])} →



Concatenation (+) Operator

Ordered union of set interpretations

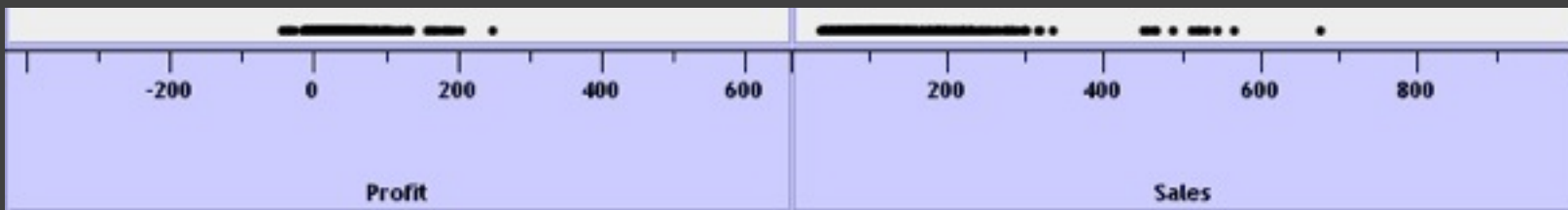
Quarter + Product Type

= {(Qtr1),(Qtr2),(Qtr3),(Qtr4)} + {(Coffee), (Espresso)}

= {(Qtr1),(Qtr2),(Qtr3),(Qtr4),(Coffee),(Espresso)}

Qtr1	Qtr2	Qtr3	Qtr4	Coffee	Espresso
48	59	57	53	151	21

Profit + Sales = {(Profit[-310,620]),(Sales[0,1000])}



Cross (x) Operator

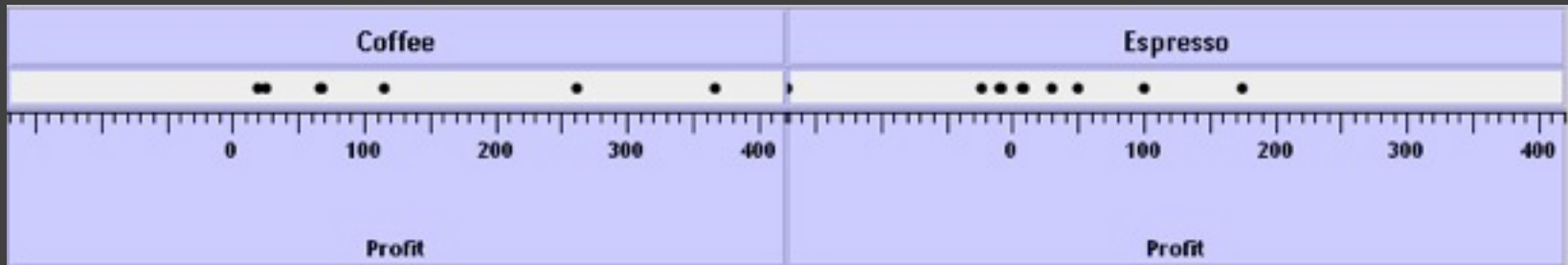
Cross-product of set interpretations

Quarter x Product Type

= {(Qtr1,Coffee), (Qtr1, Tea), (Qtr2, Coffee), (Qtr2, Tea), (Qtr3, Coffee), (Qtr3, Tea), (Qtr4, Coffee), (Qtr4,Tea)}

Qtr1		Qtr2		Qtr3		Qtr4	
Coffee	Espresso	Coffee	Espresso	Coffee	Espresso	Coffee	Espresso
131	19	160	20	178	12	134	33

Product Type x Profit =



Nest (/) Operator

Cross-product filtered by existing records

Quarter x Month

creates twelve entries for each quarter. i.e.,
(Qtr1, December)

Quarter / Month

creates three entries per quarter based on
tuples in database (not semantics)

Table Algebra

The operators (+, x, /) and operands (O, Q) provide an *algebra* for tabular visualization.

Algebraic statements are then mapped to:

Queries - selection, projection, group-by aggregation

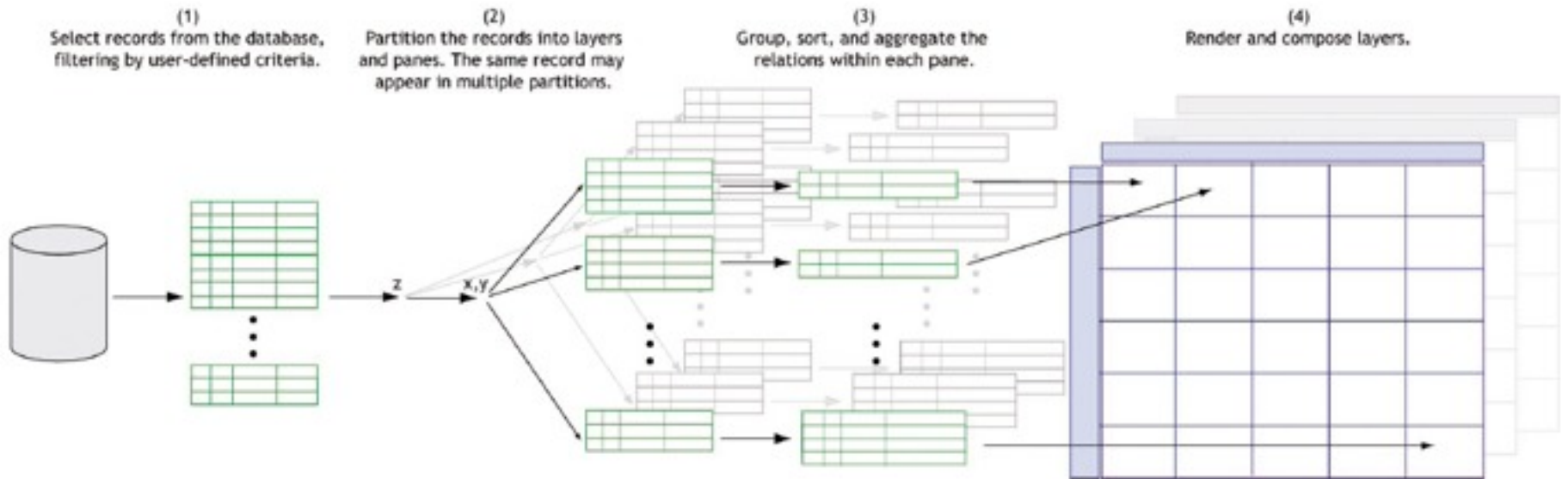
Visualizations - trellis plot partitions, visual encodings

In Tableau, users make statements via drag-and-drop

Note that this specifies operands NOT operators!

Operators are inferred by data type (O, Q)

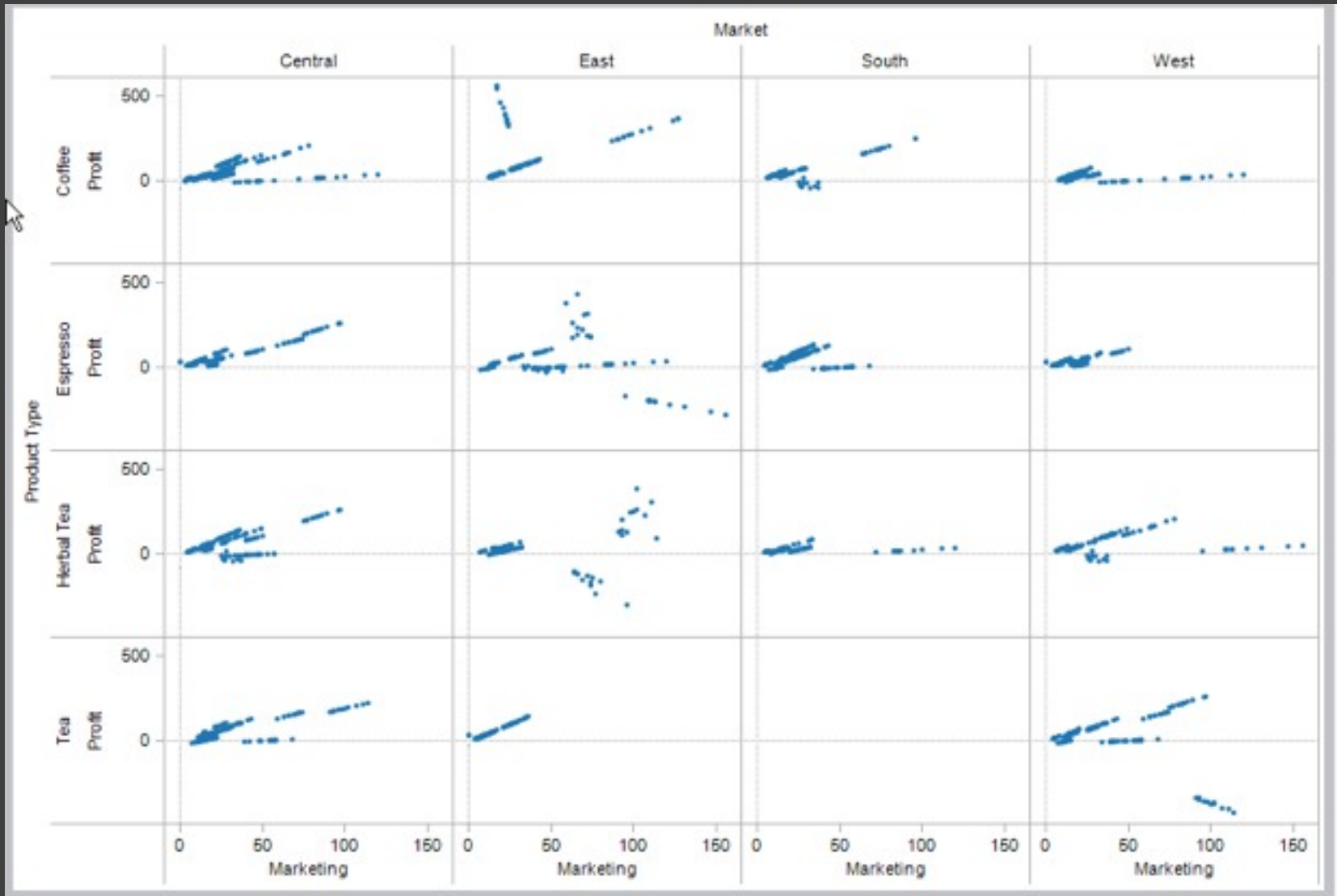
Querying the Database



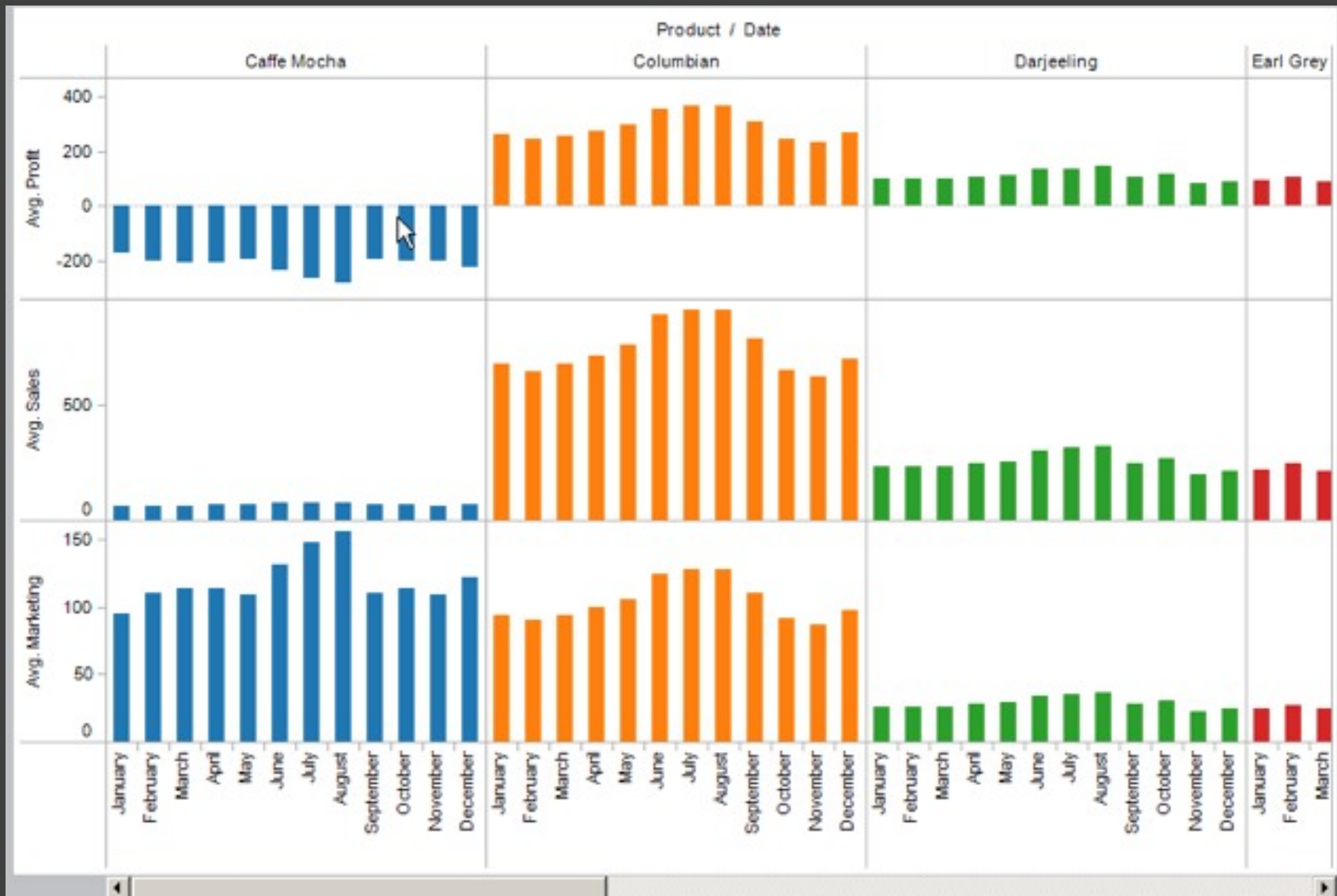
Ordinal - Ordinal

State	Product Type			
	Coffee	Espresso	Herbal Tea	Tea
Colorado	●	●	●	●
Connecticut	●	●	●	●
Florida	●	●	●	●
Illinois	●	●	●	●
Iowa	●	●	●	●
Louisiana	●	●	●	●
Massachusetts	●	●	●	●
Missouri	●	●	●	●
Nevada	●	●	●	●
New Hampshire	●	●	●	●
New Mexico	●	●	●	●
New York	●	●	●	●
Ohio	●	●	●	●
Oklahoma	●	●	●	●
Oregon	●	●	●	●
Texas	●	●	●	●
Utah	●	●	●	●
Washington	●	●	●	●
Wisconsin	●	●	●	●

Quantitative - Quantitative



Ordinal - Quantitative



Statistics and Computing

Leland Wilkinson

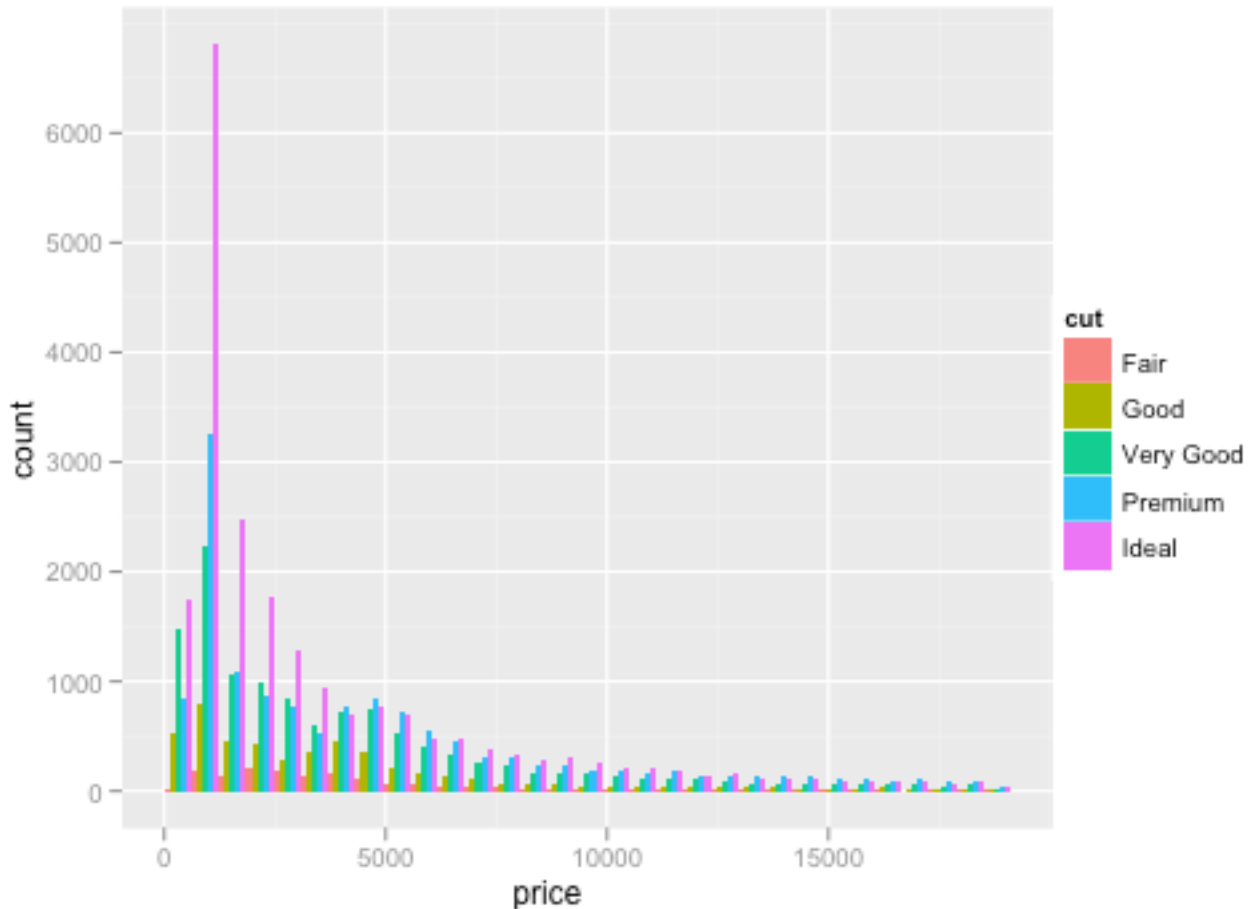
**The Grammar
of Graphics**

Second Edition

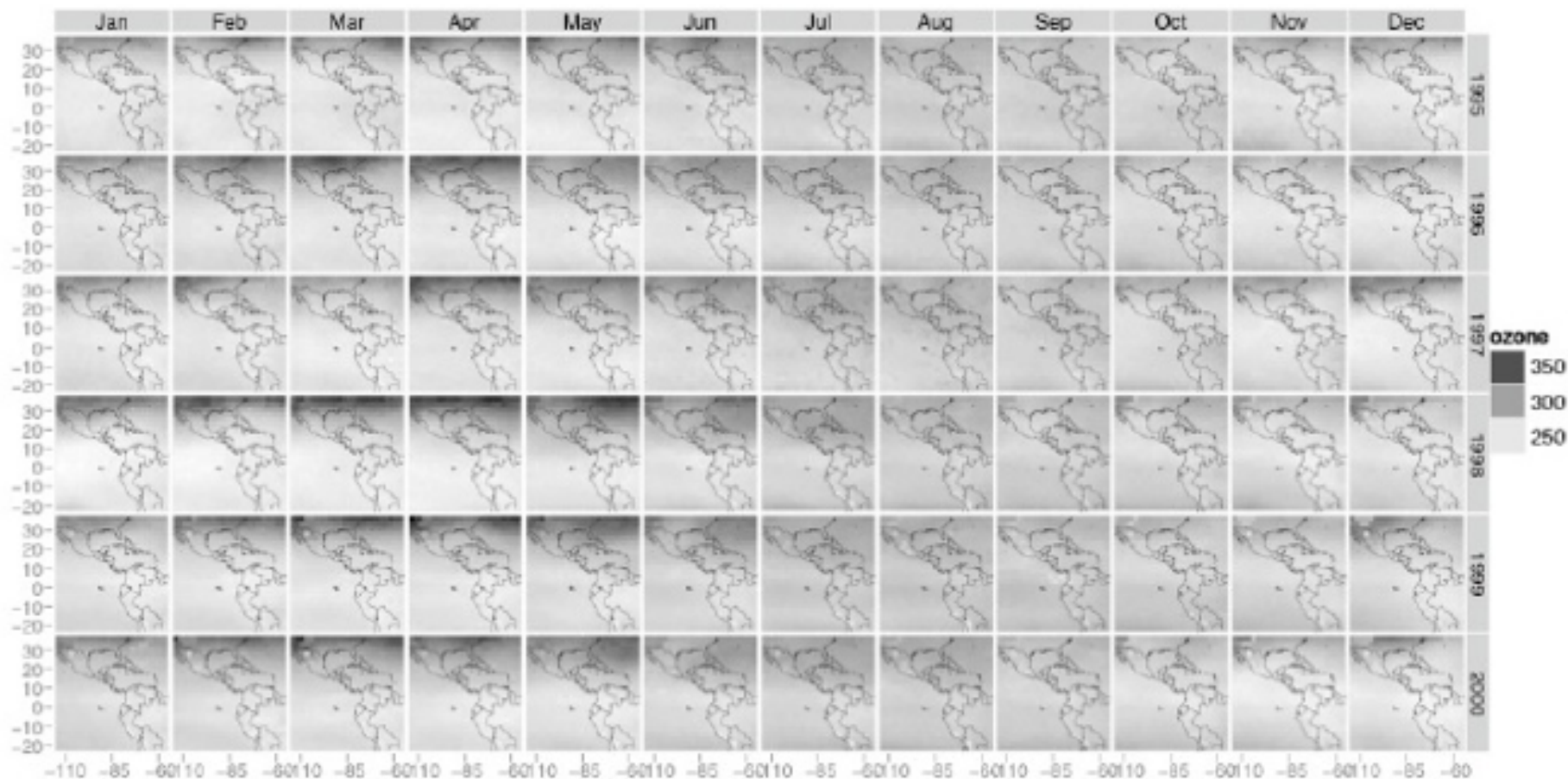
 Springer

Grammar of Graphics [Wilkinson, Wickham]

Data	Input data to visualize
Operators	Grouping, statistics, layout
Coordinates	Cartesian & polar coordinates
Scales	Map data values to visual values
Guides	Axes & legends visualize scales
Marks	Geometric primitives



```
ggplot(diamonds, aes(x=price,  
fill=cut)) + geom_bar(position="dodge")
```



```

qplot(long, lat, data = expo, geom = "tile", fill = ozone,
  facets = year ~ month) +
scale_fill_gradient(low = "white", high = "black") + map

```

Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Languages

Tableau VizQL, ggplot2

Component Model Architectures

Prefuse, Flare, Improvise

Graphics APIs

OpenGL, Java2D, GDI+, Processing

Efficiency



Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Languages

Tableau VizQL, ggplot2

Component Model Architectures

Prefuse, Flare, Improvise

Graphics APIs

OpenGL, Java2D, GDI+, Processing

Efficiency



Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Languages

Tableau VizQL, ggplot2

Component Model Architectures

Prefuse, Flare, Improvise

Graphics APIs

OpenGL, Java2D, GDI+, Processing

Expressiveness



Efficiency



Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Languages

Tableau VizQL, ggplot2



Component Model Architectures

Prefuse, Flare, Improvise

Graphics APIs

OpenGL, Java2D, GDI+, Processing

Expressiveness



Efficiency



Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Languages

Tableau VizQL, ggplot2

Visualization Grammars

Protovis, D3.js, Vega

Component Model Architectures

Prefuse, Flare, Improvise

Graphics APIs

OpenGL, Java2D, GDI+, Processing

Expressiveness

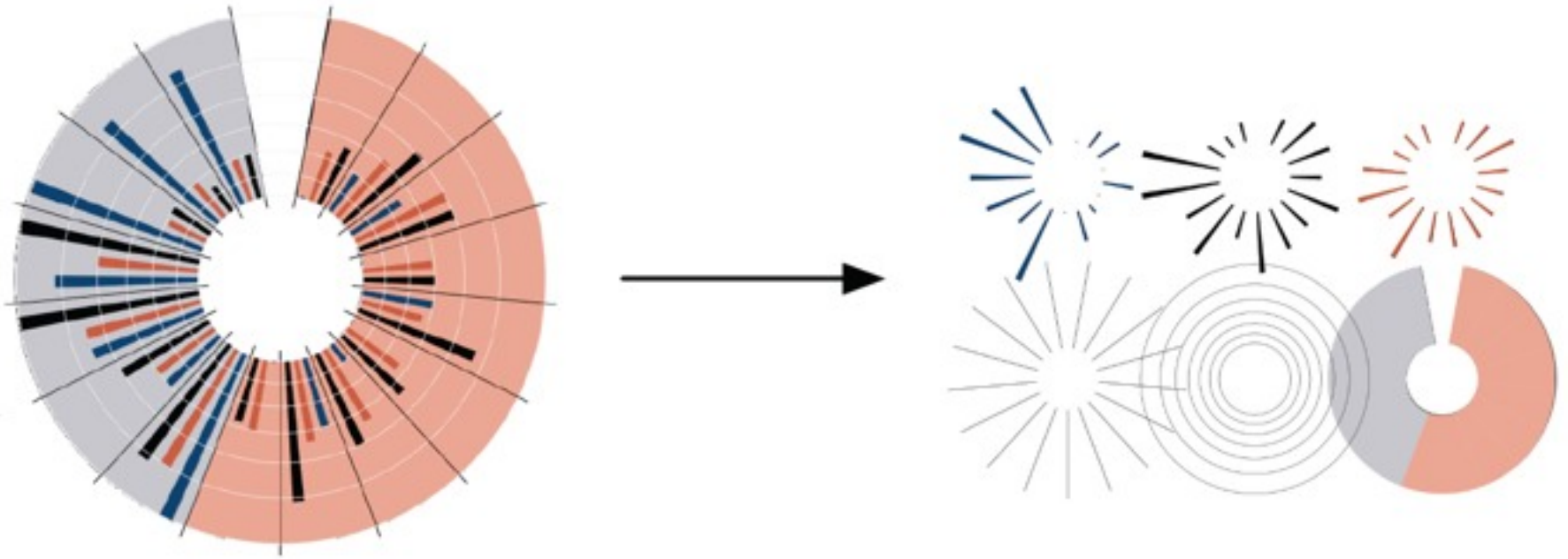


Protovis & D3

Today's first task is not to invent wholly new [graphical] techniques, though these are needed. Rather we need most vitally to recognize and reorganize the **essential of old techniques, to make easy their assembly in new ways, and to modify their external appearances to fit the new opportunities.**

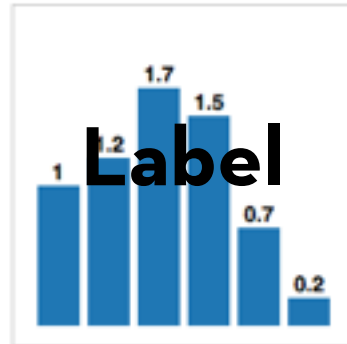
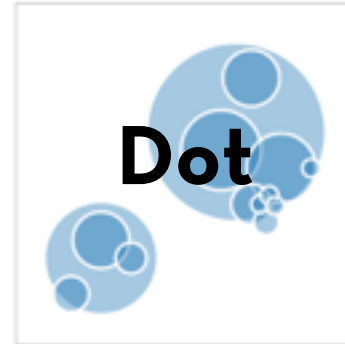
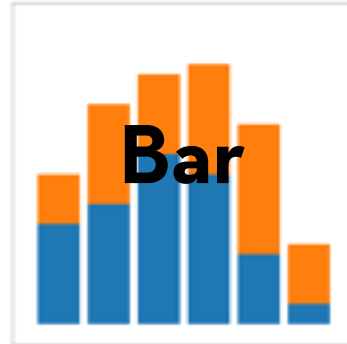
J. W. Tukey, *The Future of Data Analysis*, 1962.

Protovis: A Language for Visualization



A graphic is a composition of data-representative marks.

with **Mike Bostock** & **Vadim Ogievetsky**

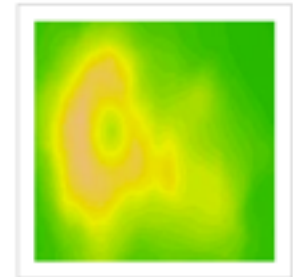


MARKS: Protovis graphical primitives

MARK

$$\lambda: D \rightarrow R$$

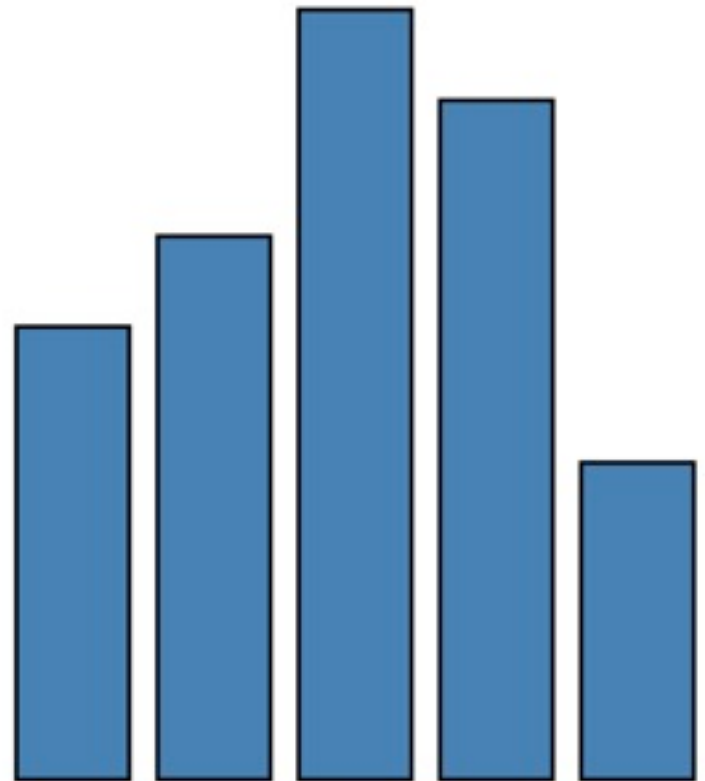
data	λ
visible	λ
left	λ
bottom	λ
width	λ
height	λ
fillStyle	λ
strokeStyle	λ
lineWidth	λ
...	λ



BAR

$$\lambda: D \rightarrow R$$

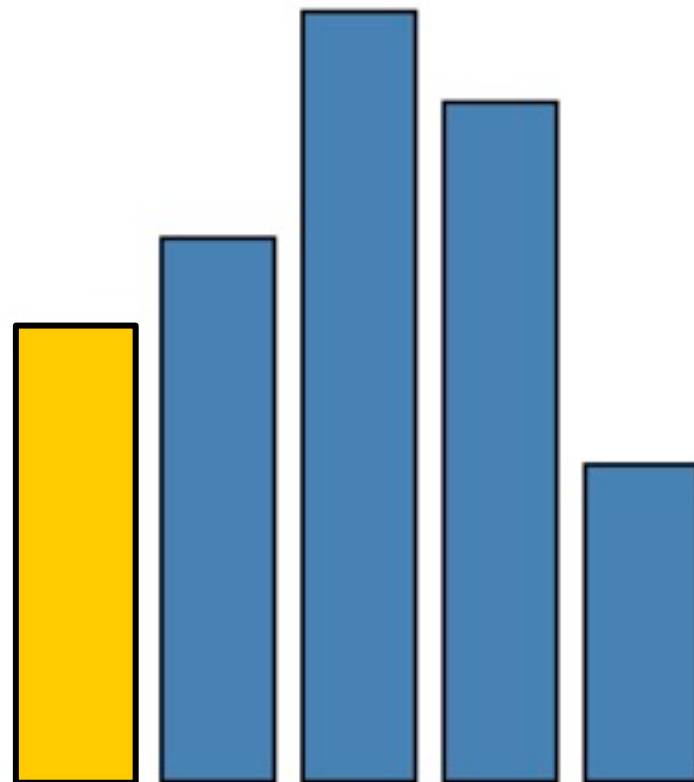
data	1	1.2	1.7	1.5	0.7
visible	true				
left	$\lambda: \text{index} * 25$				
bottom	0				
width	20				
height	$\lambda: \text{datum} * 80$				
fillStyle	blue				
strokeStyle	black				
lineWidth	1.5				
...	...				



BAR

$$\lambda: D \rightarrow R$$

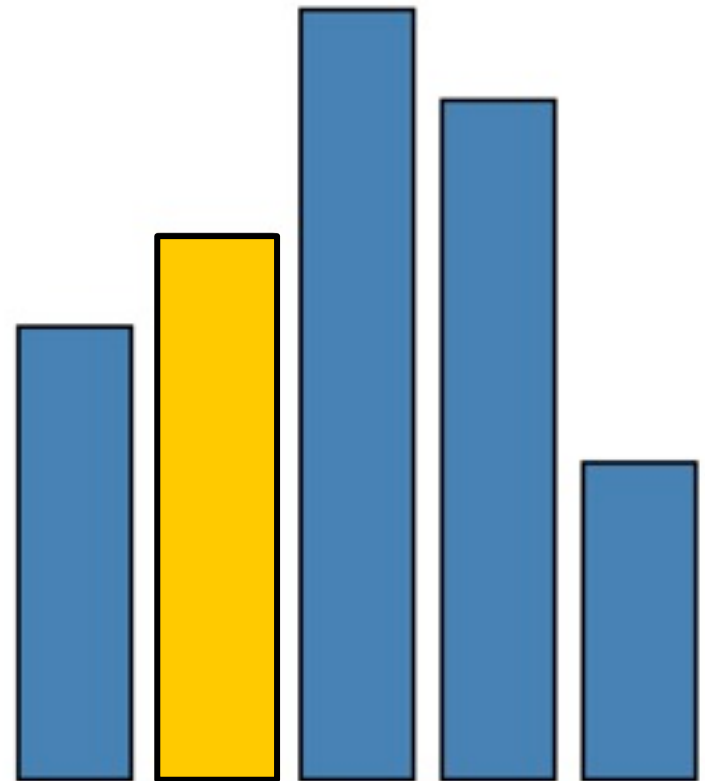
data	1	1.2	1.7	1.5	0.7
visible	true				
left	0 * 25				
bottom	0				
width	20				
height	1 * 80				
fillStyle	blue				
strokeStyle	black				
lineWidth	1.5				
...	...				



BAR

$$\lambda: D \rightarrow R$$

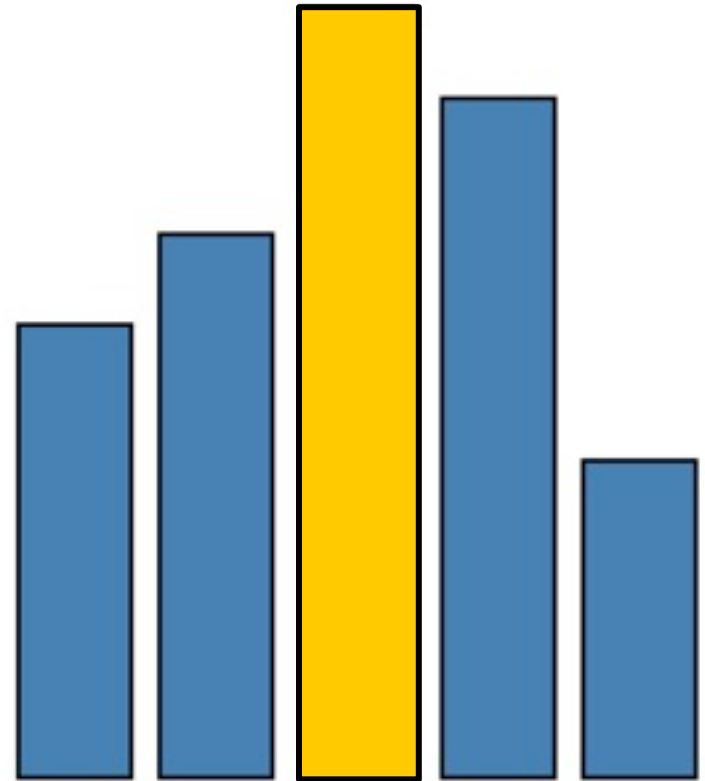
data	1	1.2	1.7	1.5	0.7
visible	true				
left	1 * 25				
bottom	0				
width	20				
height	1.2 * 80				
fillStyle	blue				
strokeStyle	black				
lineWidth	1.5				
...	...				



BAR

$$\lambda: D \rightarrow R$$

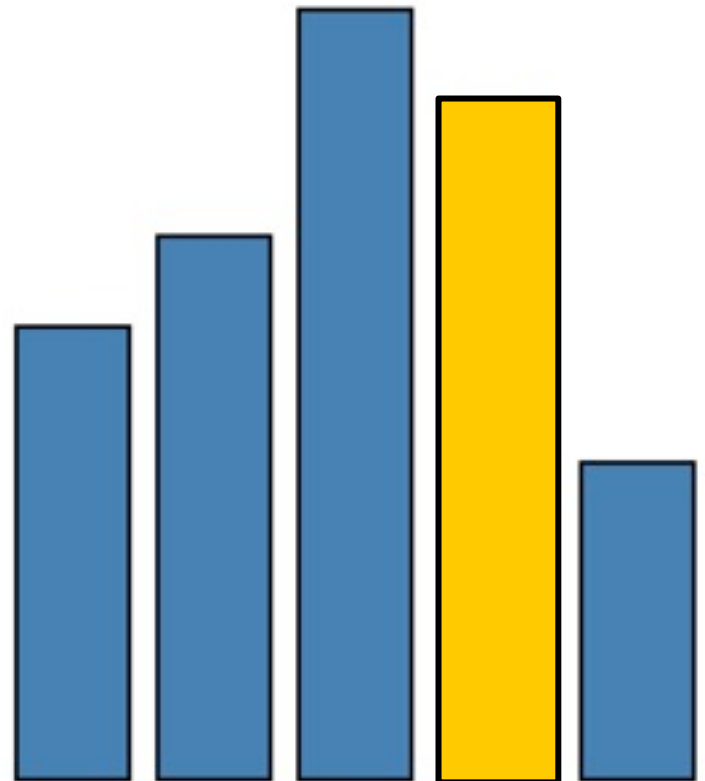
data	1	1.2	1.7	1.5	0.7
visible	true				
left	2 * 25				
bottom	0				
width	20				
height	1.7 * 80				
fillStyle	blue				
strokeStyle	black				
lineWidth	1.5				
...	...				



BAR

$$\lambda: D \rightarrow R$$

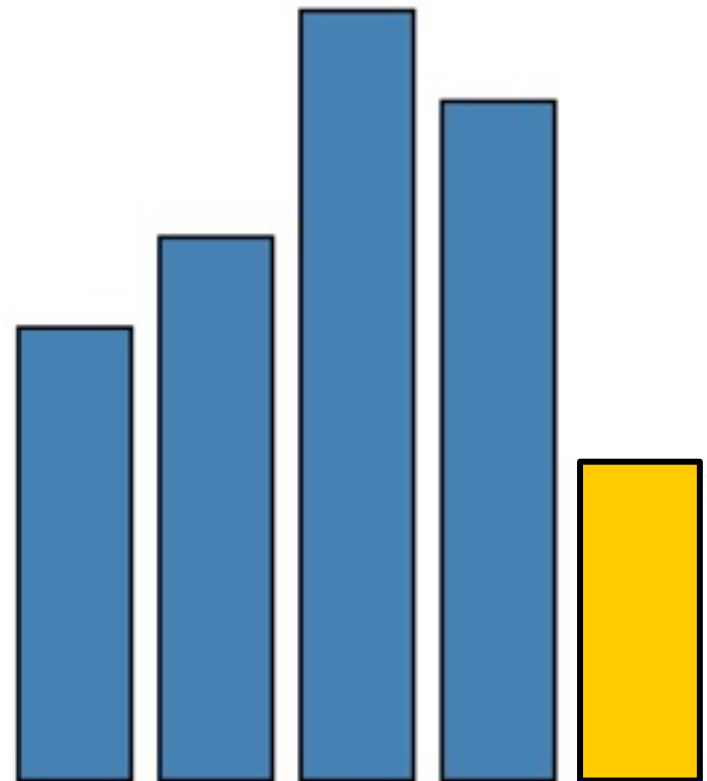
data	1	1.2	1.7	1.5	0.7
visible	true				
left	3 * 25				
bottom	0				
width	20				
height	1.5 * 80				
fillStyle	blue				
strokeStyle	black				
lineWidth	1.5				
...	...				



BAR

$$\lambda: D \rightarrow R$$

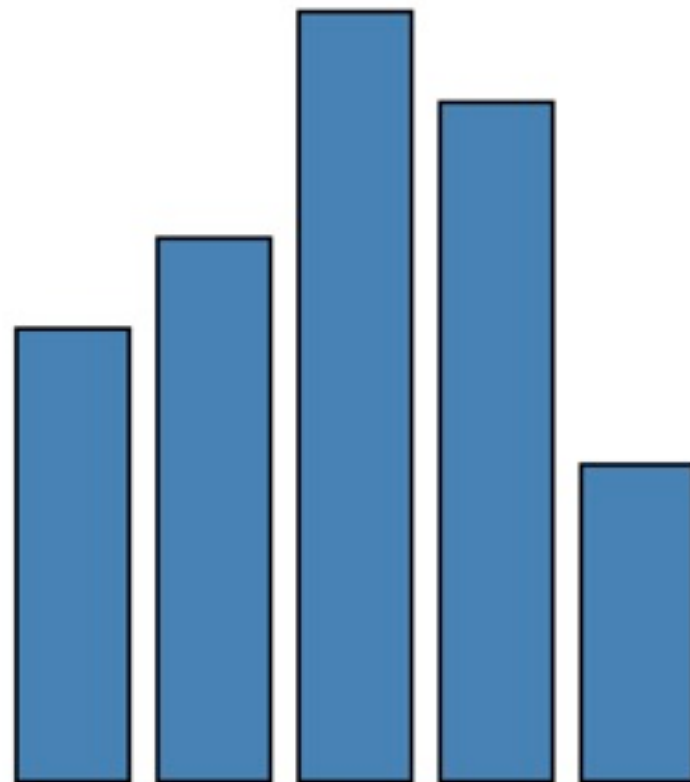
data	1	1.2	1.7	1.5	0.7
visible	true				
left	4 * 25				
bottom	0				
width	20				
height	0.7 * 80				
fillStyle	blue				
strokeStyle	black				
lineWidth	1.5				
...	...				



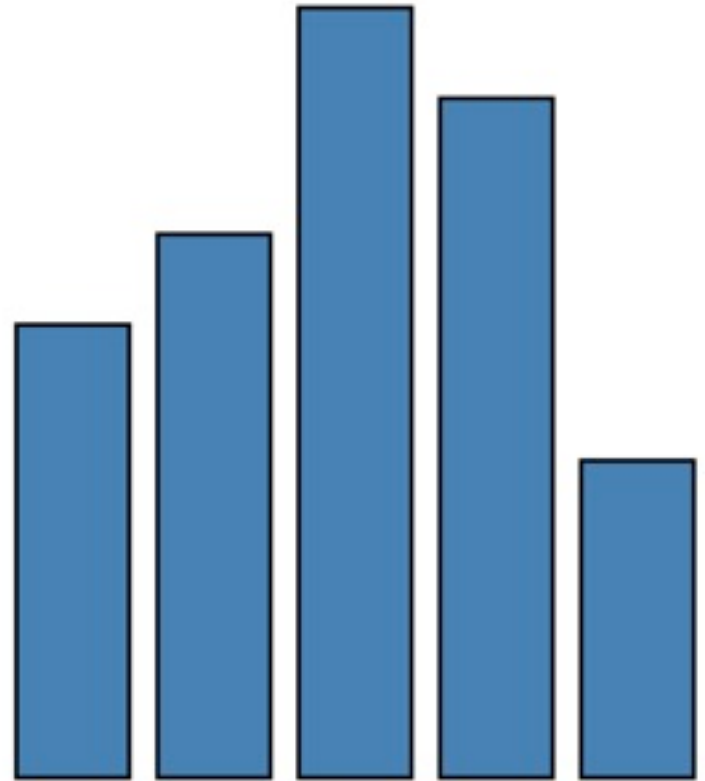
BAR

$\lambda: D \rightarrow R$

data	1	1.2	1.7	1.5	0.7
visible	true				
left	$\lambda: \text{index} * 25$				
bottom	0				
width	20				
height	$\lambda: \text{datum} * 80$				
fillStyle	blue				
strokeStyle	black				
lineWidth	1.5				
...	...				



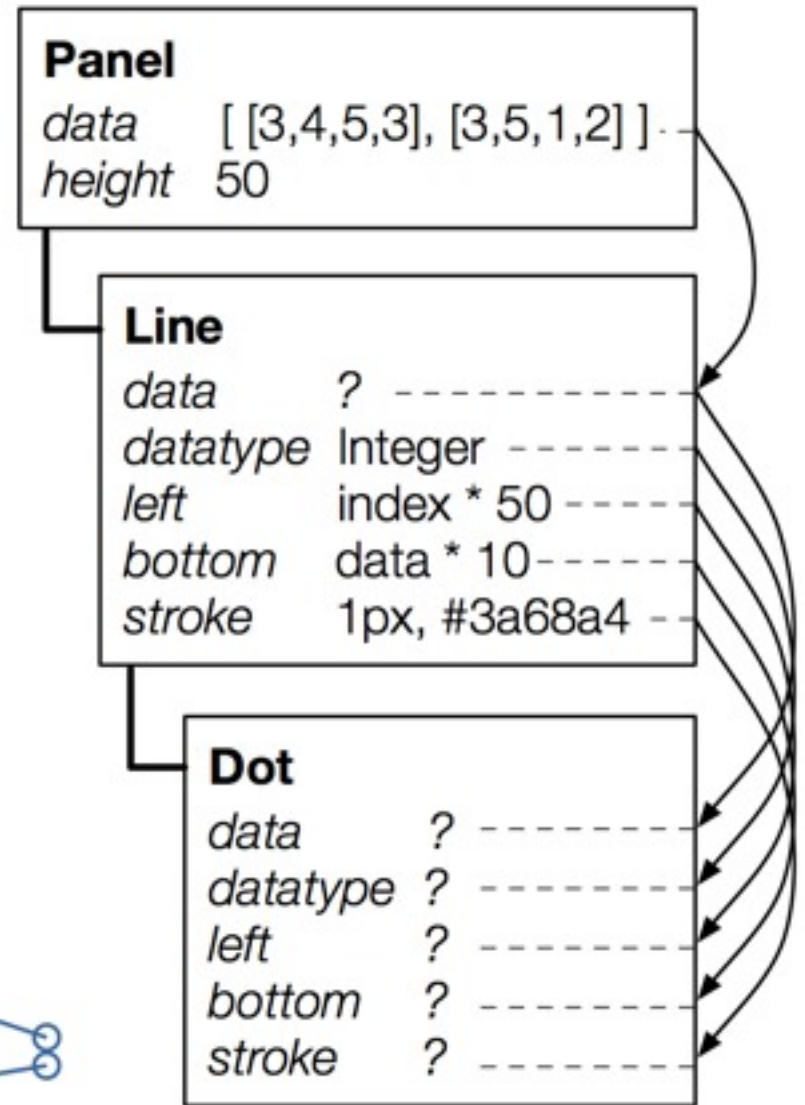
```
var vis = new pv.Panel();
vis.add(pv.Bar)
  .data([1, 1.2, 1.7, 1.5, 0.7])
  .visible(true)
  .left(function(d) this.index * 25);
  .bottom(0)
  .width(20)
  .height(function(d) d * 80)
  .fillStyle("blue")
  .strokeStyle("black")
  .lineWidth(1.5);
vis.render();
```



```

var data = [[3,4,5,3], [3,5,1,2]];
var vis = new pv.Panel()
  .data(data)
  .height(50);
vis.add(pv.Line)
  .left(function(d) this.index * 50)
  .bottom(function(d) d * 10)
  .strokeStyle("#3a68a4");
vis.add(pv.Dot);
vis.render();

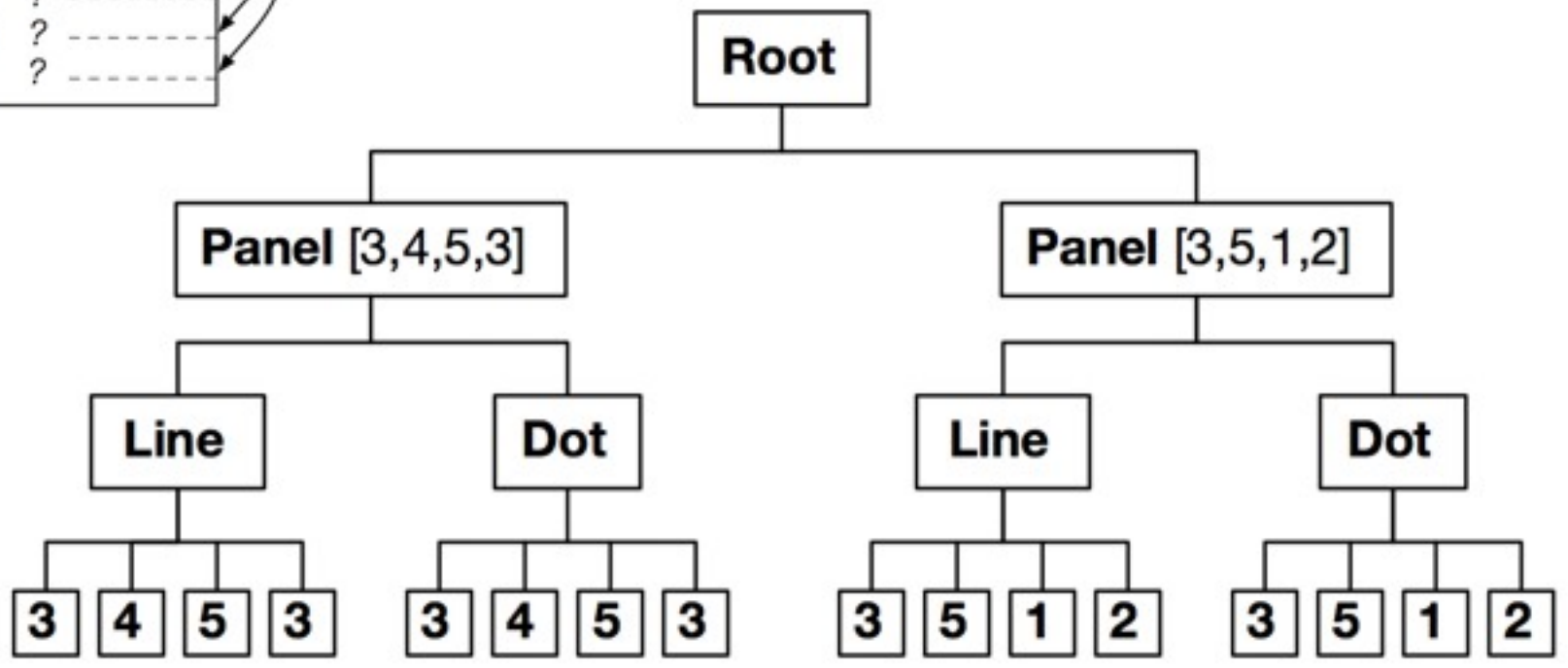
```

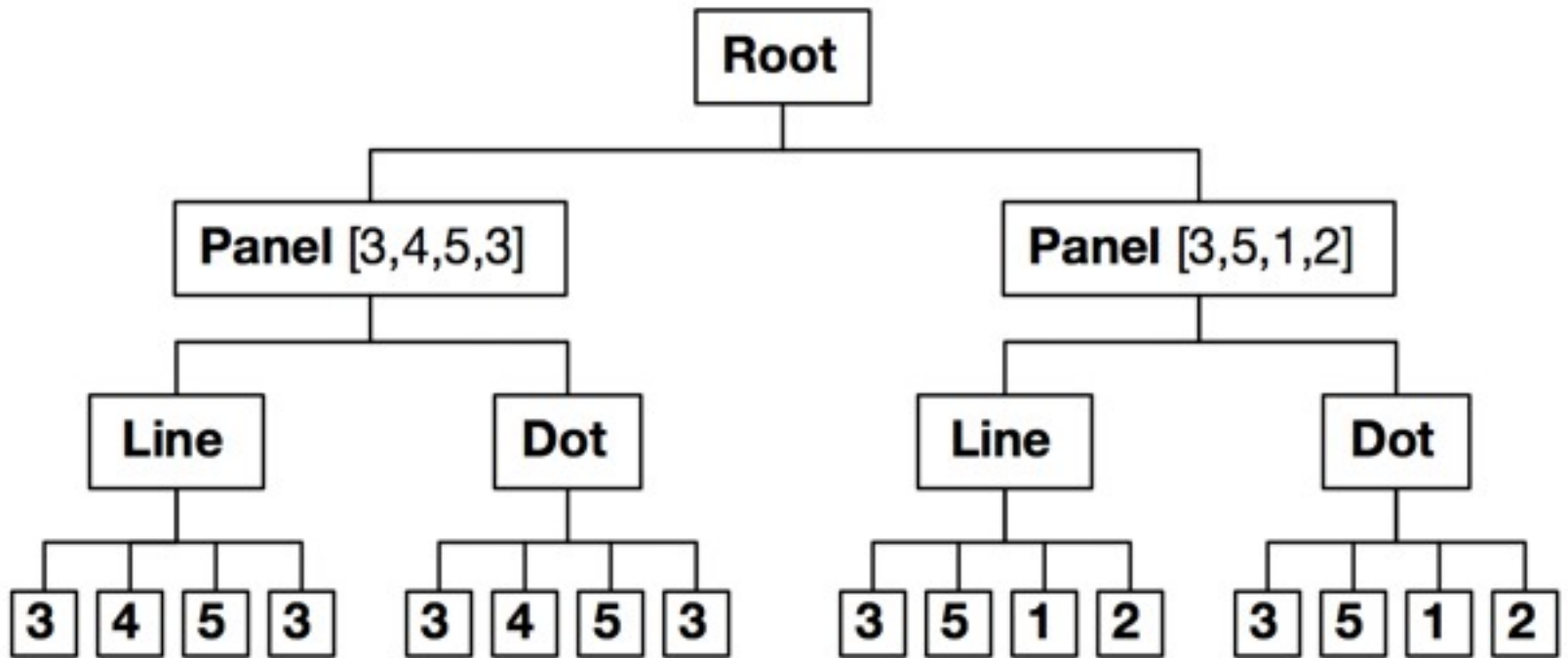


Panel
data [[3,4,5,3], [3,5,1,2]]
height 50

Line
data ?
datatype Integer
left index * 50
bottom data * 10
stroke 1px, #3a68a4

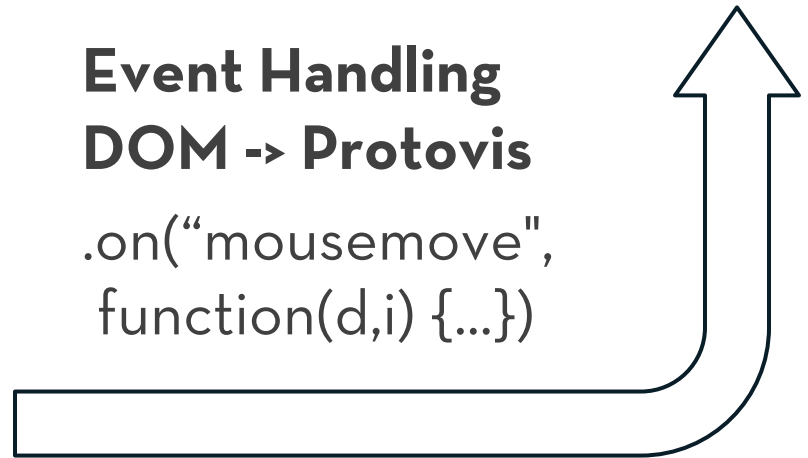
Dot
data ?
datatype ?
left ?
bottom ?
stroke ?





Event Handling DOM -> Protovis

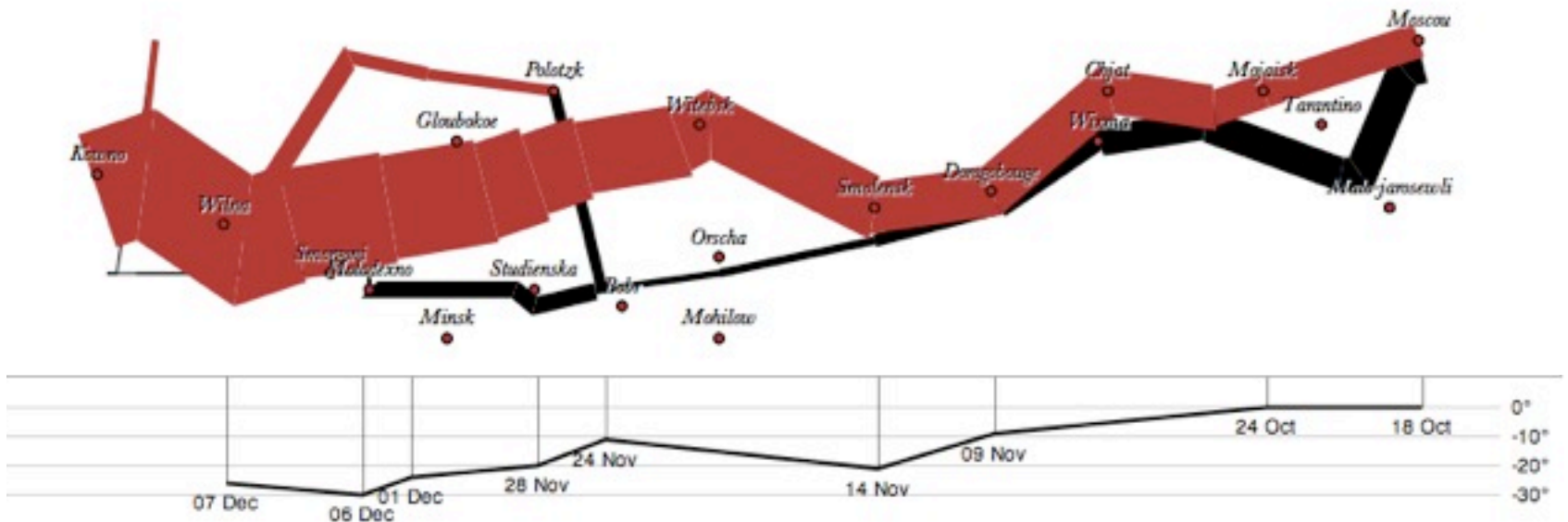
.on("mousemove",
function(d,i) {...})



Productivity - *Faster Design Cycle, Less Code*
Comparison: 5x less code, 10x less dev time

Portability - *Multiple Implementations*
JavaScript, Adobe Flash, Java/JVM

Performance - *Optimization* (in Protovis-Java)
Just-in-time compilation; parallel execution
Hardware accelerated rendering
Up to **20x scalability boost** over prior toolkits



```
var army = pv.nest(napoleon.army, "dir", "group");
var vis = new pv.Panel();
```

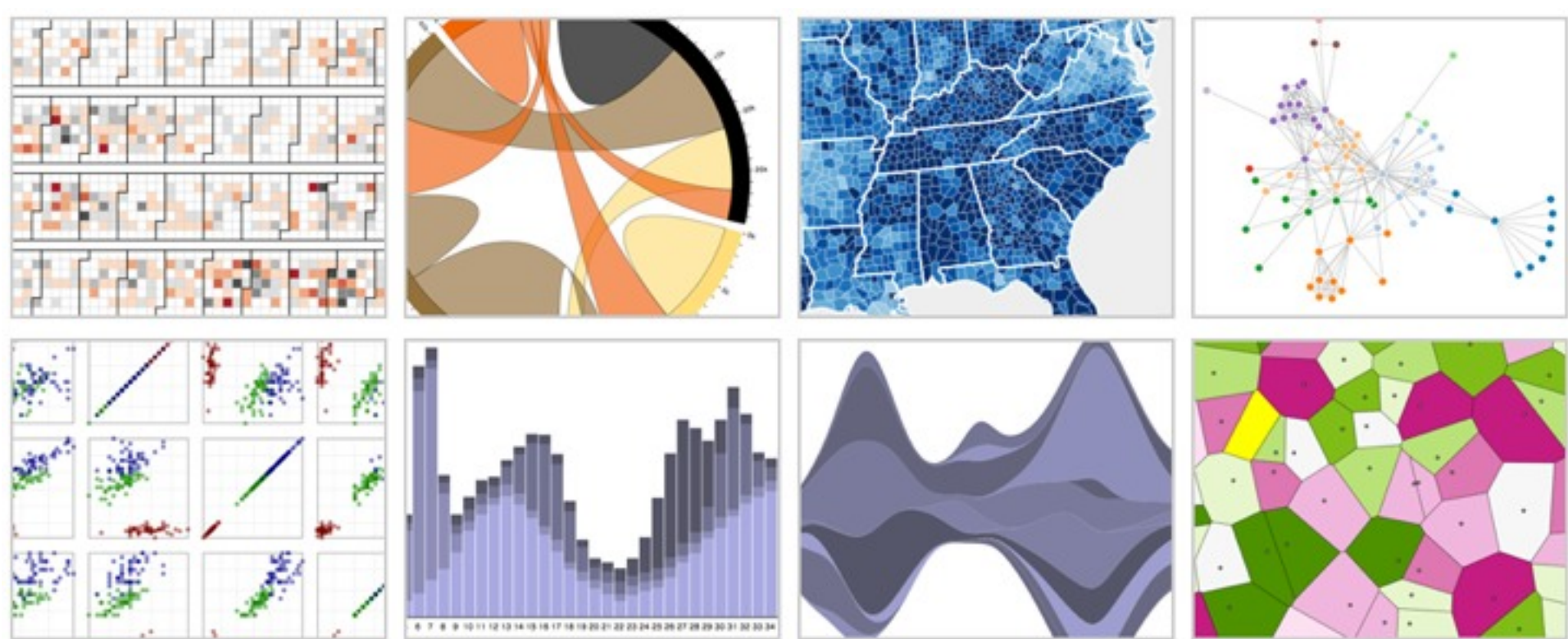
```
var lines = vis.add(pv.Panel).data(army);
lines.add(pv.Line)
  .data(function() army[this.idx])
  .left(lon).top(lat).size(function(d) d.size/8000)
  .strokeStyle(function() color[army[panelIndex]
    [0].dir]);
```

```
vis.add(pv.Label).data(napoleon.cities)
  .left(lon).top(lat)
  .text(function(d) d.city).font("italic 10px Georgia")
  .textAlign("center").textBaseline("middle");
```

```
vis.add(pv.Rule).data([0,-10,-20,-30])
  .top(function(d) 300 - 2*d - 0.5).left(200).right(150)
  .lineWidth(1).strokeStyle("#ccc")
  .anchor("right").add(pv.Label)
  .font("italic 10px Georgia")
  .text(function(d) d+"°").textBaseline("center");
```

```
vis.add(pv.Line).data(napoleon.temp)
  .left(lon).top(tmp) .strokeStyle("#0")
  .add(pv.Label)
  .top(function(d) 5 + tmp(d))
  .text(function(d) d.temp+"° "+d.date.substr(0,6))
  .textBaseline("top").font("italic 10px Georgia");
```

d3.js Data-Driven Documents



with Mike Bostock & Vadim Ogievetsky

Protovis

Specialized mark types

- + Streamlined design
- Limits expressiveness
- More overhead (slower)
- Harder to debug
- Self-contained model

Specify a scene (nouns)

- + Quick for static vis
- Delayed evaluation
- Animation, interaction
are more cumbersome

Protovis

Specialized mark types

- + Streamlined design
- Limits expressiveness
- More overhead (slower)
- Harder to debug
- Self-contained model

Specify a scene (nouns)

- + Quick for static vis
- Delayed evaluation
- Animation, interaction are more cumbersome

D3

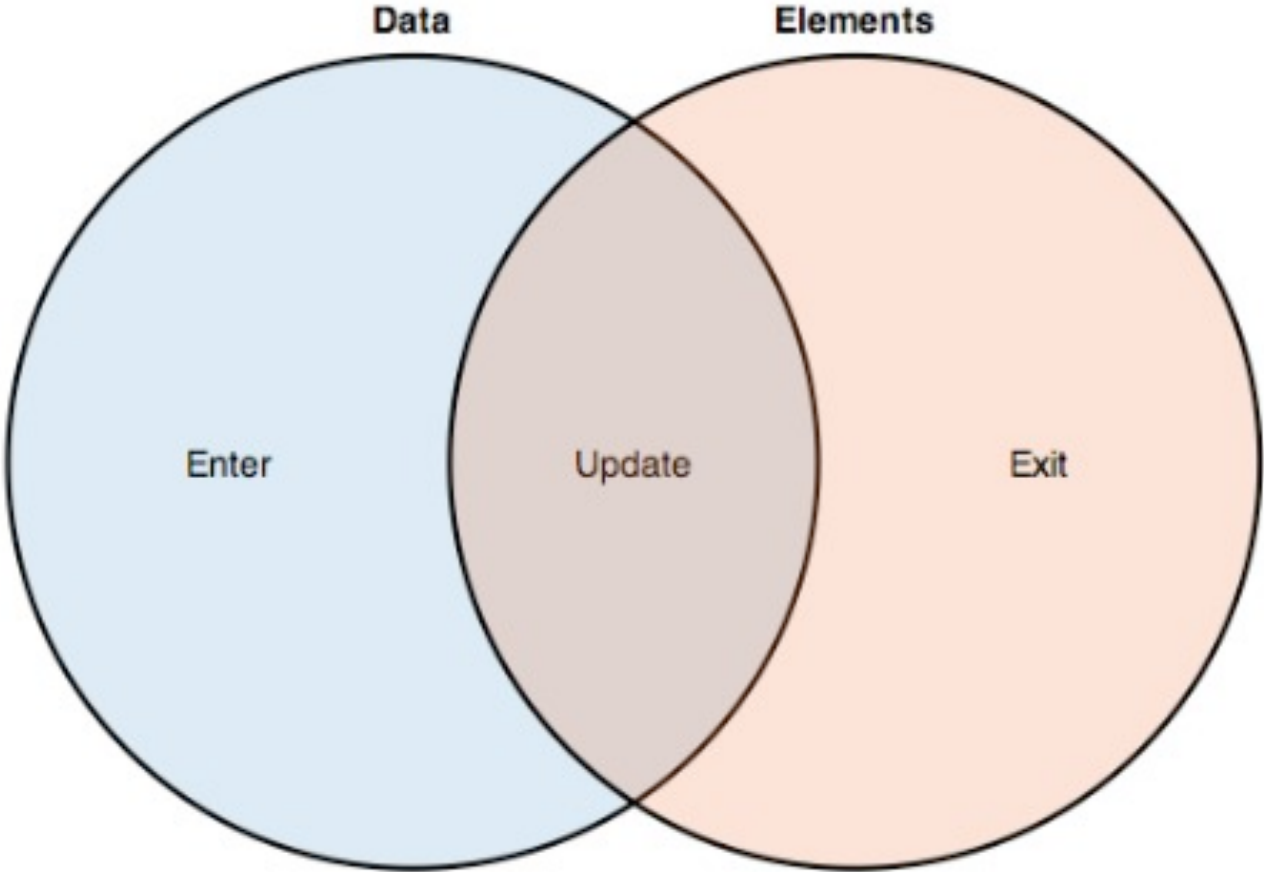
Bind data to DOM

- Exposes SVG/CSS/...
- + Exposes SVG/CSS/...
- + Less overhead (faster)
- + Debug in browser
- + Use with other tools

Transform a scene (verbs)

- More complex model
- + Immediate evaluation
- + Dynamic data, animation, and interaction natural

Selection + Data Join



Efficiency



Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Languages

Tableau VizQL, ggplot2

Visualization Grammars

Protovis, D3.js, Vega

Component Model Architectures

Prefuse, Flare, Improvise

Graphics APIs

OpenGL, Java2D, GDI+, Processing

Expressiveness



Administrivia

Assignment 2: Exploratory Data Analysis

Use visualization software to form & answer questions

First steps:

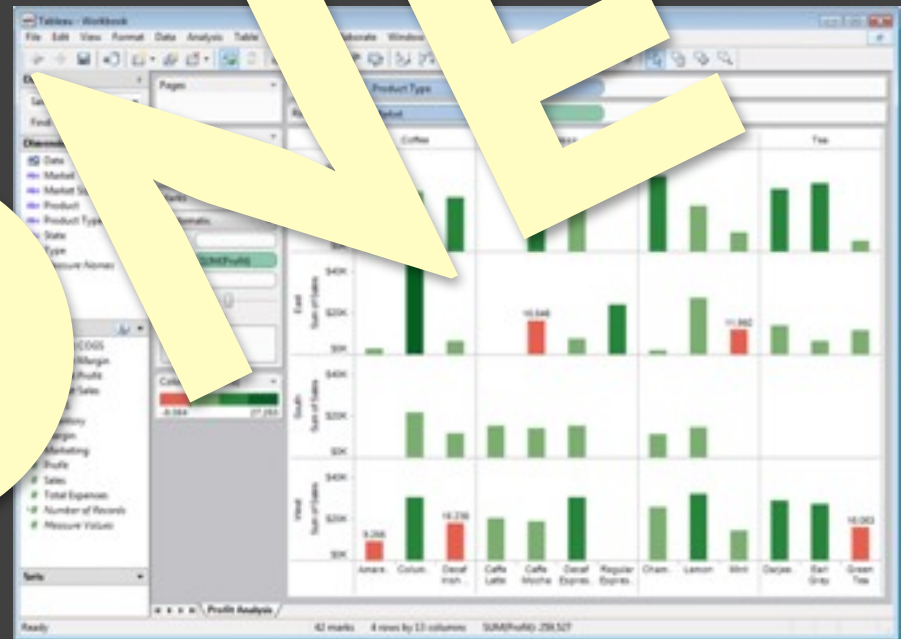
- Step 1: Pick domain & data
- Step 2: Pose questions
- Step 3: Profile the data
- Iterate as needed

Create visualization

- Interact with data
- Refine your questions

Make a notebook

- Keep records of your analysis
- Prepare a final graphic and caption



Due by 5:00pm
Monday, Jan 27

Comparing Approaches

No Humble Pie



Pie Chart: Processing

```
size(150, 150); background(255);
smooth(); noStroke();
int diameter = 140;
float[] data = {1, 1.2, 1.7, 1.5, .7};
float lastAng = 0, sum = 0;
for (int i = 0; i < data.length; i++)
    sum += data[i];
for (int i = 0; i < data.length; i++) {
    fill(data[i] * 120);
    float ang = data[i] / sum * 2 * PI;
    arc(width / 2, height / 2, diameter, diameter,
        lastAng, lastAng + ang);
    lastAng += ang;
}
```

Pie Chart: Flare

```
var data:Data = Data.fromArray(
    [{value:1}, {value:1.2}, {value:1.7}, ...]
);
var vis:Visualization = new Visualization(data);

vis.bounds = new Rectangle(0, 0, 150, 150);
vis.operators.add(new PieLayout("data.value", 0.9));
vis.operators.add(new ColorEncoder(
    "data.value", "nodes", "fillColor"));

vis.update();
addChild(vis);
```

Pie Chart: Protovis

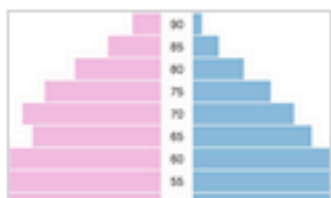
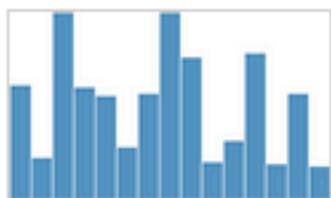
```
var vis = new pv.Panel()
    .width(150)
    .height(150)
    .add(pv.Wedge)
    .left(75)
    .bottom(75)
    .outerRadius(70)
    .angle(function(d) { return d * 2 * Math.PI; })
    .data(pv.normalize([1, 1.2, 1.7, 1.5, .7]));

vis.render();
```

Pie Chart: D3

```
var data = [1, 1.2, 1.7, 1.5, .7];
var color = d3.scale.category20();
var svg = d3.select("body").append("svg")
  .attr("width", 150)
  .attr("height", 150);
var g = svg.append("g")
  .attr("transform", "translate(75, 75)");
var arcs = g.selectAll("path")
  .data(d3.layout.pie().sort(null)(data))
  .enter().append("path")
  .style("fill", function(d,i) { return color(i); })
  .attr("d", d3.svg.arc().innerRadius(0).outerRadius(70));
```

Vega & Lyra



vega

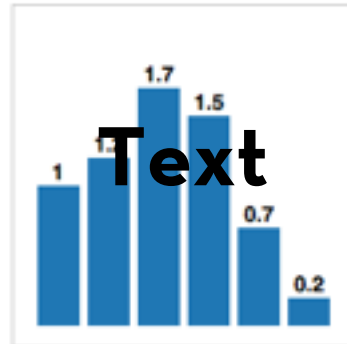
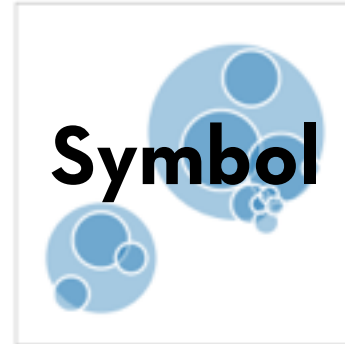
vega.js (127k)
vega.min.js (63k)
Source (GitHub)

Vega is a visualization grammar, a declarative format for creating, saving and sharing visualization designs.

With Vega you can describe data visualizations in a JSON format, and generate interactive views using either HTML5 Canvas or SVG.

Read the [tutorial](#), browse the [documentation](#), and then explore Vega visualizations using the web-based [Vega Editor](#).

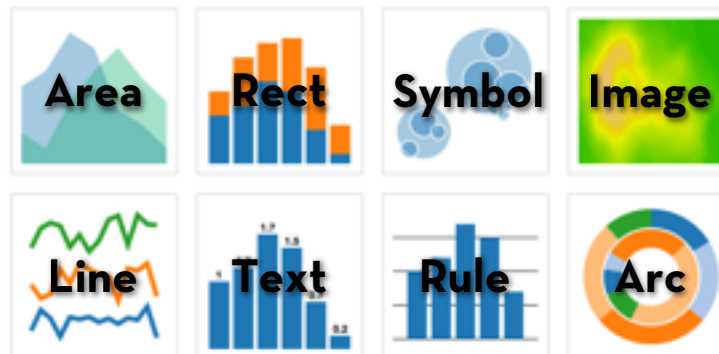




MARKS: Vega graphical primitives

Vega Grammar

Data	Input data to visualize
Transforms	Grouping, stats, projection, layout
Scales	Map data values to visual values
Guides	Axes & legends visualize scales
Marks	Data-representative graphics



```

{
  "name": "pie", "width": 150, "height": 150,
  "data": [{
    "name": "table",
    "values": [1, 1.2, 1.7, 1.5, 0.7],
    "transform": [{"type": "pie", "value": "data"}]
  }],
  "scales": [
    {"name": "color", "type": "ordinal", "range": "category20"}
  ],
  "marks": [{
    "type": "arc",
    "from": {"data": "table"},
    "properties": {
      "enter": {
        "x": {"group": "width", "mult": 0.5},
        "y": {"group": "height", "mult": 0.5},
        "startAngle": {"field": "startAngle"},
        "endAngle": {"field": "endAngle"},
        "innerRadius": {"value": 0},
        "outerRadius": {"value": 70},
        "fill": {"scale": "color", "field": "index"}
      }
    }
  ]
}

```

Vega Design Goals

Balancing rapid + expressive design

Protovis-style marks, data transforms & layouts

Reuse & sharing of visualization designs

JSON format -> chart components or stand-alone vis

Support programmatic generation

Provides an intermediate specification language

Performance & platform flexibility

Browser or server-side; Canvas or SVG

For the highest level graphics (elegant, custom, expensive), enter the crunched data or the graphical output into **Adobe Illustrator**. ...This program gives complete control over typography, line weight, color, grids, layout--just what we need for doing graphical work.

It is a serious, complex design program; you may want to work with **real graphical designers** who will surely know their way around Illustrator.

Tufte, E. *Ask E.T.: Graphics Software*, April 2001

Lyra: An Interactive Visualization Design Environment

DATA PIPELINES

Full Data

Left Labels

Pipeline name:

Import data from:

Filter:

side	left	left	left	left	left	left
year	1956	1974	1979	1980	1986	1989
miles	3675	5943	6744	6672	7558	8397
gas	2.38	2.34	2.68	3.3	1.76	1.75
index	0	18	23	24	30	33

1-12 of 12

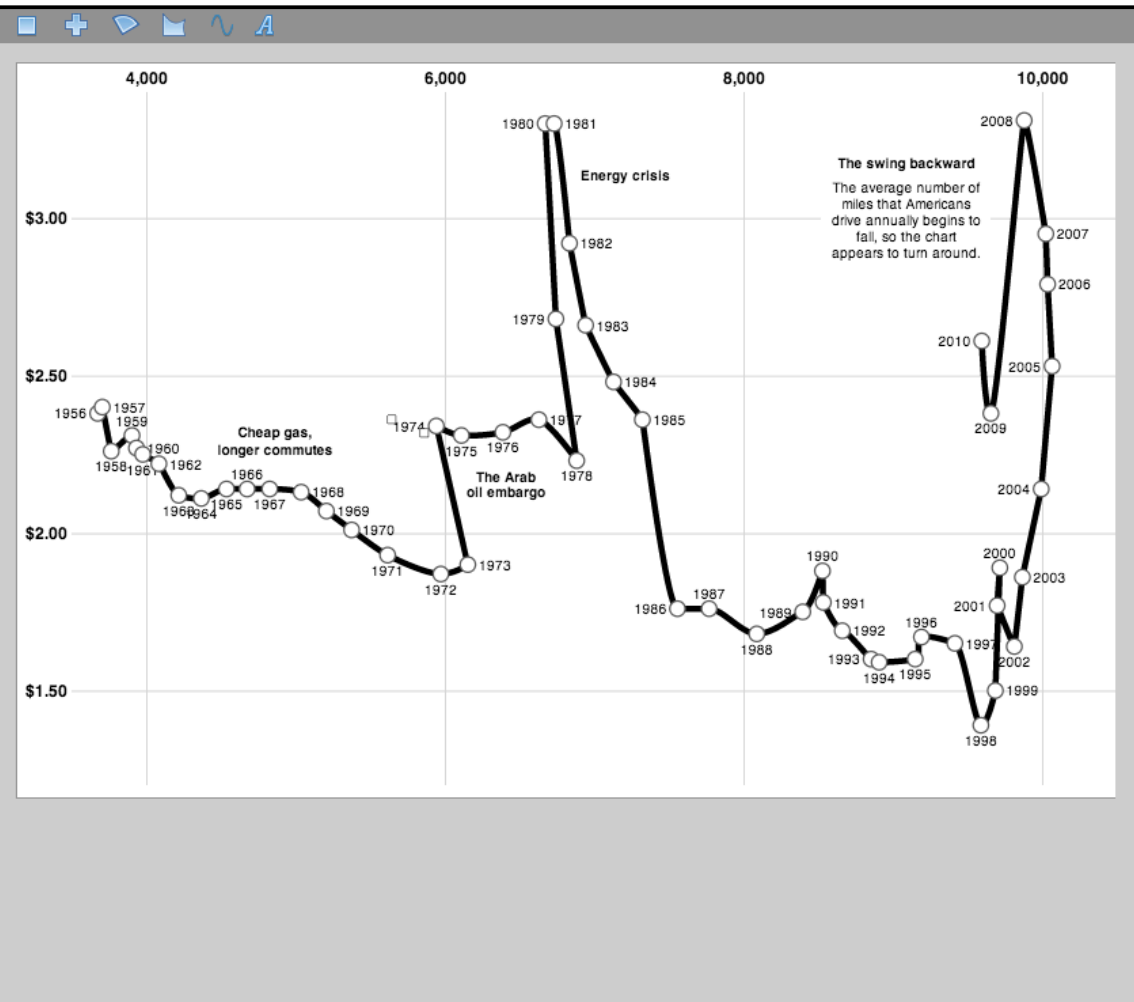
Data Transforms

Scales

Right Labels

Top Labels

Bottom Labels



VISUAL LAYERS

Plot

Line

Bottom Labels

Top Labels

Right Labels

Left Labels

name:

pipeline name:

Visual Transforms

text:

font

family:

size:

weight:

style:

color:

position

x:

y:

align

x:

y:

DATA PIPELINES

Wheat Wages

Monarchs

Pipeline name:

Import data from:

Extend
 formula: `(name == "Cromwell")`
 as:

Extend
 formula: `(! commonwealth && ind`
 as:

name	Elizabeth	James I	Charles I	Cromwell
start	1565	1603	1625	1649
end	1603	1625	1649	1660
index	0	1	2	3
commonw...	false	false	false	true
offset	0	1	0	0

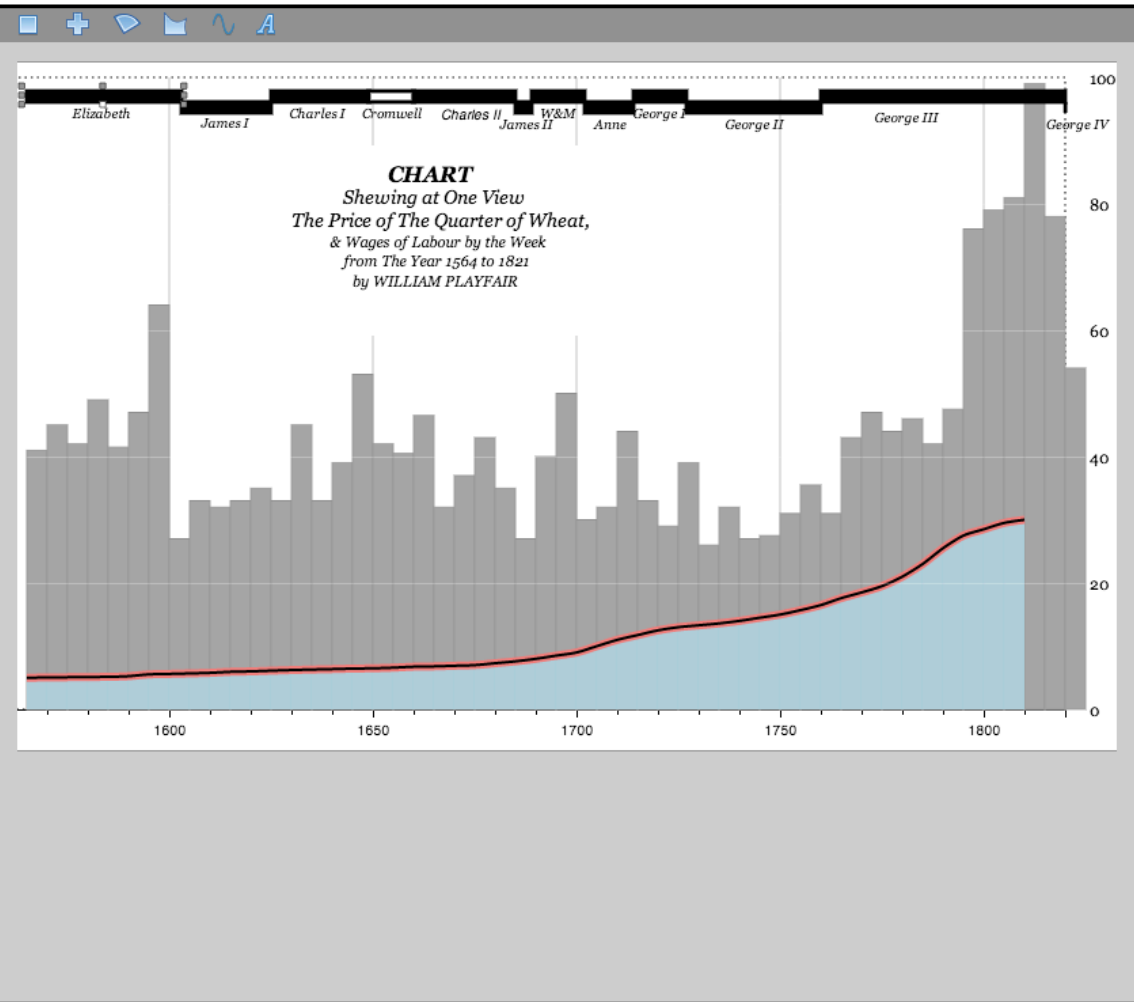
1-12 of 12

Data Transforms

Scales

LIN | X LIN | Y ORD | Fill Color

50 Years



VISUAL LAYERS

Axes

Wheat
Year

Marks

Monarchs

name:

pipeline:

Visual Transforms

x

start:

width:

end:

y

start:

height:

end:

fill

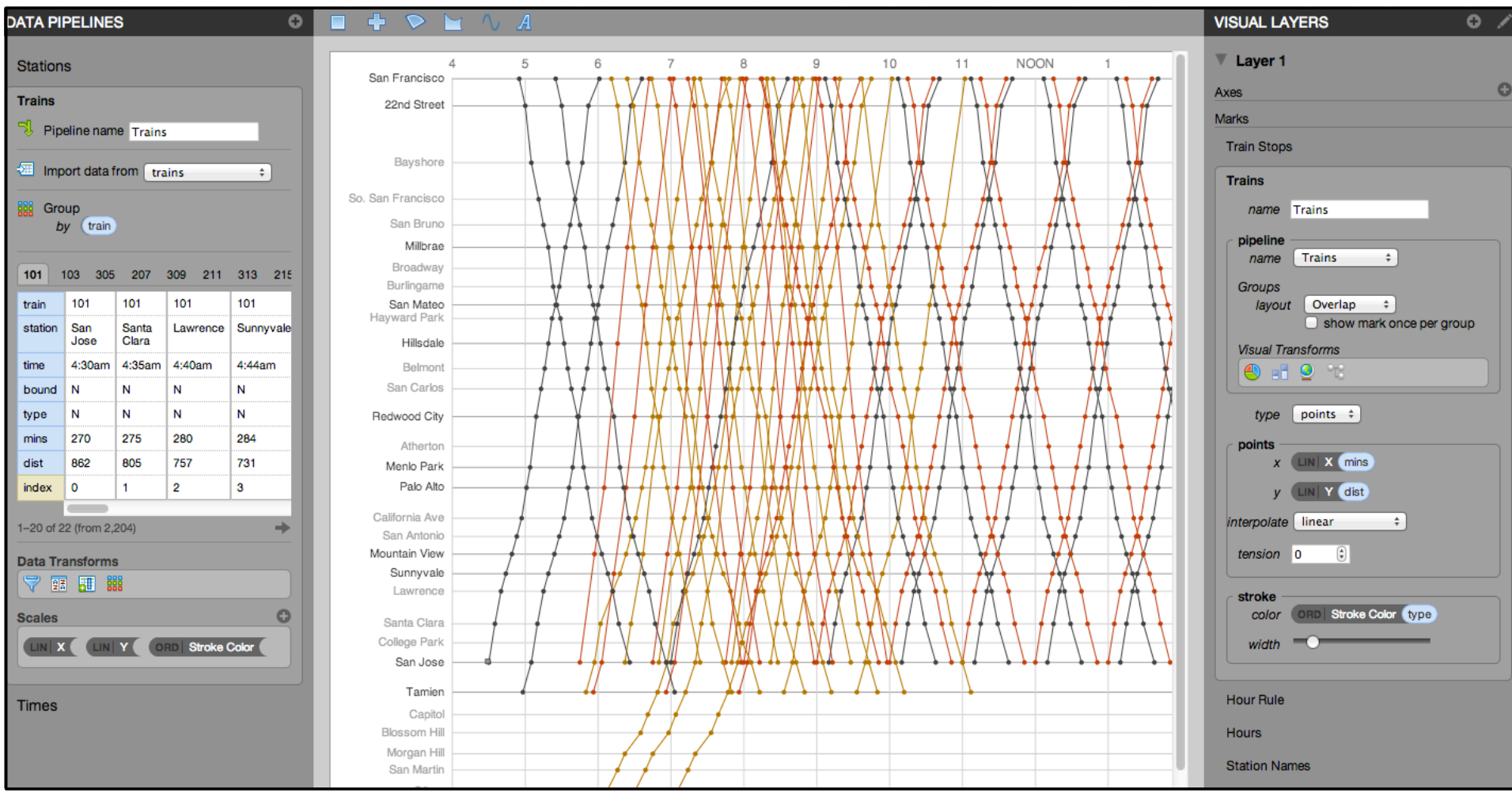
color:

opacity:

stroke

color:

width:



DATA PIPELINES

Zip Codes

Pipeline name: Zip Codes

Import data from: zipcodes-full

Group by: state

33	36	72	78	25	44	23	50	09
zip	00210	00211	00212					
lat	+43.005895	+43.005895	+43.005895					
lon	-071.013202	-071.013202	-071.013202					
code	U	U	U					
city	PORTSMOUTH	PORTSMOUTH	PORTSMOUTH					
state	33	33	33					
county	015	015	015					
index	0	1	2					
key	33	33	33					

1-20 of 284 (from 42,192)

Data Transforms

Scales

ORD | Stroke Color



VISUAL LAYERS

Visual Transforms

Geo

type: Latitude/Longitude

latitude: lat

longitude: lon

projection: mercator

center

x: -98.35

y: 39.50

translate

x: 350

y: 170

scale: 775

rotate: 0

precision: 0

clip angle: 0

output: x y

type: points

points

x: x

y: y

interpolate: monotone

tension: 0

DATA PIPELINES

Army

Pipeline name

Import data from

Group by

Window size step

1 1	1 -1	2 1	2 -1	3 1	3 -1
lon	24	24.5	25.5	26	27
lat	54.9	55	54.6	54.7	54.8
size	340000	340000	340000	320000	300
dir	1	1	1	1	1
group	1	1	1	1	1
index	0	1	2	3	4
key	1 1	1 1	1 1	1 1	1 1

1-16 of 16 (from 49)

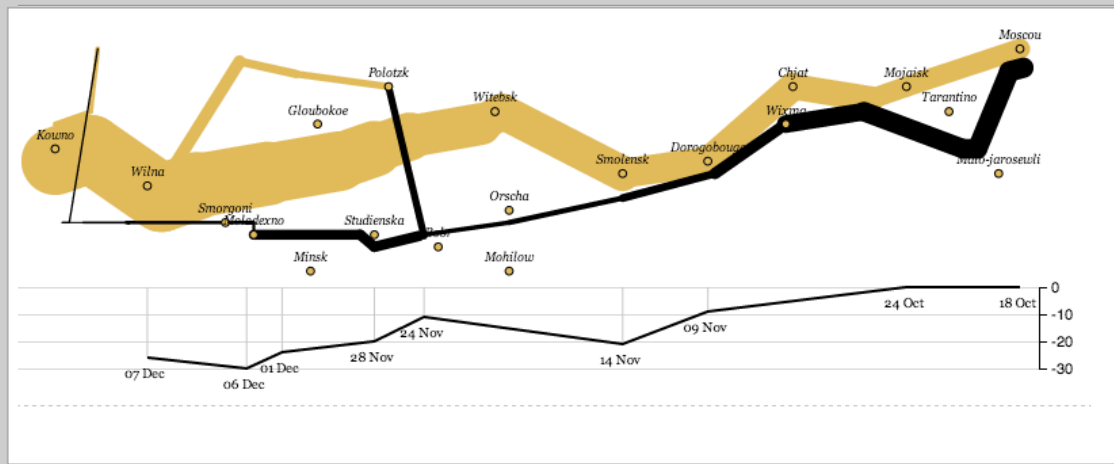
Data Transforms

Scales

ORD | Stroke Color | LIN | Stroke Width

Cities

Temperatures



VISUAL LAYERS

latitude

longitude

projection

center

x

y

translate

x

y

scale

rotate

precision

clip angle

output

type

points

x

y

interpolate

tension

stroke

color

width