Ch 7: Arrange Tables

Tamara Munzner
Department of Computer Science
University of British Columbia

CPSC 547, Information Visualization
Day 8: 26 January 2017

http://www.cs.ubc.ca/~tmm/courses/547-17
VAD Ch 7: Arrange Tables

Encode

- **Arrange**
  - Express
  - Order
  - Use
- **Separate**
- **Align**
- **Use**
Arrange tables

Express Values

Separate, Order, Align Regions

Separate
Order
Align

Axis Orientation
Rectilinear
Parallel
Radial

Layout Density
Dense
Space-Filling

1 Key
2 Keys
3 Keys
Many Keys

List
Recursive Subdivision
Matrix
Volume
Keys and values

- **key**
  - independent attribute
  - used as unique index to look up items
  - simple tables: 1 key
  - multidimensional tables: multiple keys

- **value**
  - dependent attribute, value of cell

- **classify arrangements by key count**
  - 0, 1, 2, many...

- **Express Values**
  - 1 Key
    - List
  - 2 Keys
    - Matrix
  - 3 Keys
    - Volume
  - Many Keys
    - Recursive Subdivision
Idiom: **scatterplot**

- *express values*
  - quantitative attributes
- *no keys, only values*
  - data
    - 2 quant attribs
  - mark: points
  - channels
    - horiz + vert position
- *tasks*
  - find trends, outliers, distribution, correlation, clusters
- *scalability*
  - hundreds of items

Some keys: Categorical regions

- **regions**: contiguous bounded areas distinct from each other
  - using space to *separate* (proximity)
  - following expressiveness principle for categorical attributes

- use ordered attribute to *order* and *align* regions

- 1 Key  
  - List

- 2 Keys  
  - Matrix

- 3 Keys  
  - Volume

- Many Keys  
  - Recursive Subdivision
Idiom: **bar chart**

- one key, one value
  - data
    - 1 categ attrib, 1 quant attrib
  - mark: lines
  - channels
    - length to express quant value
    - spatial regions: one per mark
      - separated horizontally, aligned vertically
      - ordered by quant attrib
        » by label (alphabetical), by length attrib (data-driven)
  - task
    - compare, lookup values
  - scalability
    - dozens to hundreds of levels for key attrib
Idiom: stacked bar chart

• one more key
  – data
    • 2 categ attrib, 1 quant attrib
  – mark: vertical stack of line marks
    • glyph: composite object, internal structure from multiple marks
  – channels
    • length and color hue
    • spatial regions: one per glyph
      – aligned: full glyph, lowest bar component
      – unaligned: other bar components
  – task
    • part-to-whole relationship
  – scalability
    • several to one dozen levels for stacked attrib

Idiom: **streamgraph**

- generalized stacked graph
  - emphasizing horizontal continuity
  - vs vertical items
- data
  - 1 categ key attrib (artist)
  - 1 ordered key attrib (time)
  - 1 quant value attrib (counts)
- derived data
  - geometry: layers, where height encodes counts
  - 1 quant attrib (layer ordering)
- scalability
  - hundreds of time keys
  - dozens to hundreds of artist keys
  - more than stacked bars, since most layers don’t extend across whole chart

Idiom: **line chart**

- one key, one value
  - data
    - 2 quant attribs
  - mark: points
    - line connection marks between them
  - channels
    - aligned lengths to express quant value
    - separated and ordered by key attrib into horizontal regions
- task
  - find trend
    - connection marks emphasize ordering of items along key axis by explicitly showing relationship between one item and the next
Choosing bar vs line charts

- depends on type of key attrib
  - bar charts if categorical
  - line charts if ordered
- do not use line charts for categorical key attribs
  - violates expressiveness principle
    - implication of trend so strong that it overrides semantics!
      - “The more male a person is, the taller he/she is”

Idiom: heatmap

- two keys, one value
  - data
    - 2 categ attrs (gene, experimental condition)
    - 1 quant attrib (expression levels)
  - marks: area
    - separate and align in 2D matrix
      - indexed by 2 categorical attributes
  - channels
    - color by quant attrib
      - (ordered diverging colormap)
  - task
    - find clusters, outliers
- scalability
  - 1M items, 100s of category levels, ~10 quant attrib levels
Idiom: cluster heatmap

• in addition
  – derived data
    • 2 cluster hierarchies
  – dendrogram
    • parent-child relationships in tree with connection line marks
    • leaves aligned so interior branch heights easy to compare
  – heatmap
    • marks (re-)ordered by cluster hierarchy traversal
Axis Orientation

- Rectilinear
- Parallel
- Radial
Idioms: **scatterplot matrix, parallel coordinates**

- **scatterplot matrix (SPLOM)**
  - rectilinear axes, point mark
  - all possible pairs of axes
  - scalability
    - one dozen attribs
    - dozens to hundreds of items

- **parallel coordinates**
  - parallel axes, jagged line representing item
  - rectilinear axes, item as point
    - axis ordering is major challenge
  - scalability
    - dozens of attribs
    - hundreds of items

---

*after [Visualization Course Figures. McGuffin, 2014.](http://www.michaelmcguffin.com/courses/vis/)*
Task: Correlation

• scatterplot matrix
  – positive correlation
    • diagonal low-to-high
  – negative correlation
    • diagonal high-to-low
  – uncorrelated

• parallel coordinates
  – positive correlation
    • parallel line segments
  – negative correlation
    • all segments cross at halfway point
  – uncorrelated
    • scattered crossings


Idioms: radial bar chart, star plot

• radial bar chart
  – radial axes meet at central ring, line mark

• star plot
  – radial axes, meet at central point, line mark

• bar chart
  – rectilinear axes, aligned vertically

• accuracy
  – length unaligned with radial
    • less accurate than aligned with rectilinear

Idioms: **pie chart, polar area chart**

- **pie chart**
  - area marks with angle channel
  - accuracy: angle/area much less accurate than line length

- **polar area chart**
  - area marks with length channel
  - more direct analog to bar charts

- **data**
  - 1 categ key attrib, 1 quant value attrib

- **task**
  - part-to-whole judgements

Idioms: **normalized stacked bar chart**

- task
  - part-to-whole judgements

- normalized stacked bar chart
  - stacked bar chart, normalized to full vert height
  - single stacked bar equivalent to full pie
    - high information density: requires narrow rectangle

- pie chart
  - information density: requires large circle

http://bl.ocks.org/mbostock/3887235
http://bl.ocks.org/mbostock/3886208
http://bl.ocks.org/mbostock/3886394
Idiom: **glyphmaps**

- rectilinear good for linear vs nonlinear trends

- radial good for cyclic patterns

Orientation limitations

- rectilinear: scalability wrt #axes
  - 2 axes best
  - 3 problematic
    - more in afternoon
  - 4+ impossible
- parallel: unfamiliarity, training time
- radial: perceptual limits
  - angles lower precision than lengths
  - asymmetry between angle and length
  - can be exploited!

Further reading

  — *Chap 7: Arrange Tables*

Paper: D3

• paper types
  – design studies
  – technique/algorithm
  – evaluation
  – model/taxonomy
  – system

Toolkits

• imperative: how
  – low-level rendering: Processing, OpenGL
  – parametrized visual objects: prefuse
    • also flare: prefuse for Flash

• declarative: what
  – Protoviz, D3, ggplot2
  – separation of specification from execution

• considerations
  – expressiveness
    • can I build it?
  – efficiency
    • how long will it take?
  – accessibility
    • do I know how?
WebGL/OpenGL

• graphics library
  – pros
    • power and flexibility, complete control for graphics
    • hardware acceleration
    • many language bindings: C, C++, Java (w/ JOGL)
  – cons
    • big learning curve if you don’t know already
    • no vis support, must roll your own everything
  – example app: TreeJuxtaposer (OpenGL)

Processing

• layer on top of Java/OpenGL
• visualization esp. for artists/designers
• pros
  – great sandbox for rapid prototyping
  – huge user community, great documentation
• cons
  – poor widget library support
• example app: MizBee

prefuse

• infovis toolkit, in Java
• fine-grained building blocks for tailored visualizations
• pros
  – heavily used (previously)
  – very powerful abstractions
  – quickly implement most techniques covered so far
• cons
  – no longer active
  – nontrivial learning curve
• example app: DOITrees Revisited

prefuse

- separation: abstract data, visual form, view
  - data: tables, networks
  - visual form: layout, color, size, ...
  - view: multiple renderers

InfoVis Reference Model

• conceptual model underneath design of prefuse and many other toolkits
• heavily influenced much of infovis (including nested model)
  —aka infovis pipeline, data state model

[Redrawn Fig 1.23. Card, Mackinlay, and Shneiderman. Readings in Information Visualization: Using Vision To Think, Chapter 1. Morgan Kaufmann, 1999.]
Declarative toolkits

• imperative tools/libraries
  – say exactly how to do it
  – familiar programming model
    • OpenGL, prefuse, ...

• declarative: other possibility
  – just say what to do
  – Protovis, D3
Protovis

• declarative infovis toolkit, in Javascript
  – also later Java version
• marks with inherited properties
• pros
  – runs in browser
  – matches mark/channel mental model
  – also much more: interaction, geospatial, trees,…
• cons
  – not all kinds of operations supported
• example app: NapkinVis (2009 course project)

[Fig 1, 3. Chao. NapkinVis. http://www.cs.ubc.ca/~tmm/courses/533-09/projects.html#will]
Protovis Validation

• wide set of old/new app examples
  – expressiveness, effectiveness, scalability
  – accessibility

• analysis with cognitive dimensions of notation
  – closeness of mapping, hidden dependencies
  – role-expressiveness visibility, consistency
  – viscosity, diffuseness, abstraction
  – hard mental operations

• declarative infovis toolkit, in Javascript
• Protovis meets Document Object Model
• pros
  – seamless interoperability with Web
  – explicit transforms of scene with dependency info
  – massive user community, many thirdparty apps/libraries on top of it, lots of docs
• cons
  – even more different from traditional programming model
• example apps: many
D3

• objectives
  – compatibility
  – debugging
  – performance

• related work typology
  – document transformers
  – graphics libraries
  – infovis systems
    • general note: all related work sections are a mini-taxonomy!

D3 capabilities

• query-driven selection
  – selection: filtered set of elements queries from the current doc
    • also partitioning/grouping!
  – operators act on selections to modify content
    • instantaneous or via animated transitions with attribute/style interpolators
    • event handlers for interaction

• data binding to scenegraph elements
  – data joins bind input data to elements
  – enter, update, exit subselections
  – sticky: available for subsequent re-selection
  – sort, filter

D3 Features

• document transformation as atomic operation
  – scene changes vs representation of scenes themselves

• immediate property evaluation semantics
  – avoid confusing consequences of delayed evaluation

• validation
  – performance benchmarks
    • page loads, frame rate
  – accessibility
    • everybody has voted with their feet by now!
Next Time

• to read
  – VAD Ch. 8: Arrange Spatial Data
  • paper type: technique