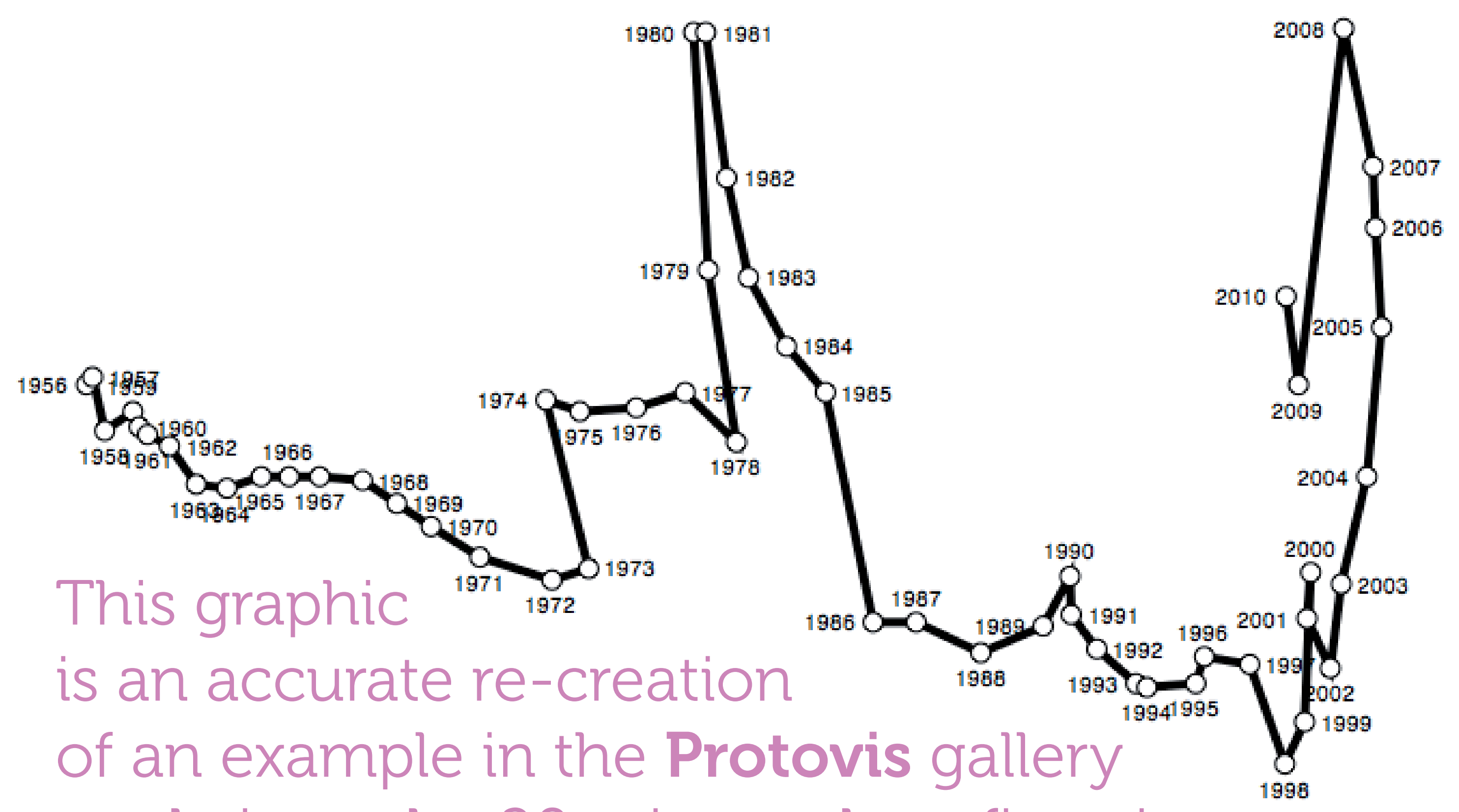


This graphic is an accurate re-creation of an example in the **Protovis** gallery made in under 20 minutes by a first-time user.

ProtoWiz

Moderately Complex Visualizations for the Ambitious Non-Programmer

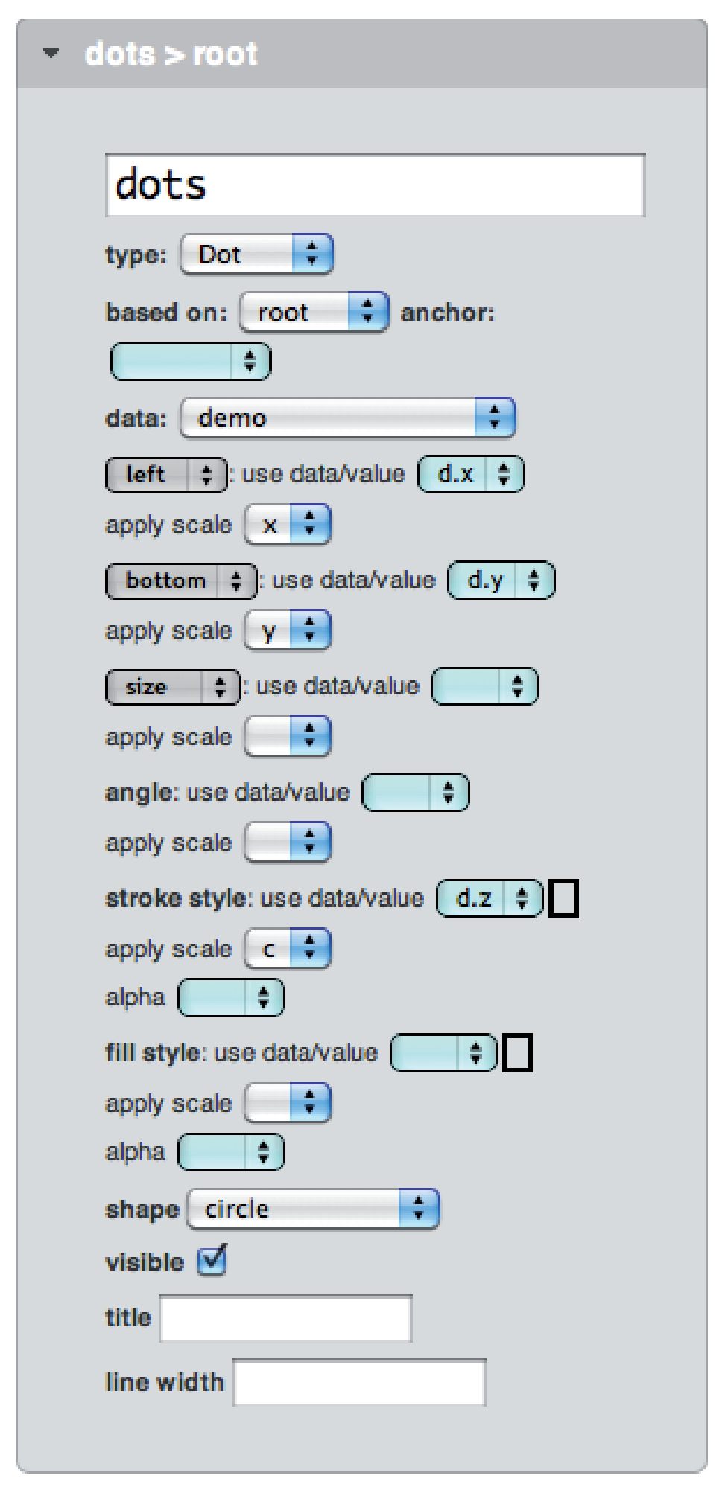
Michael Cohen, Energy and Resources Group | Thomas Schluchter, School of Information

Project Summary

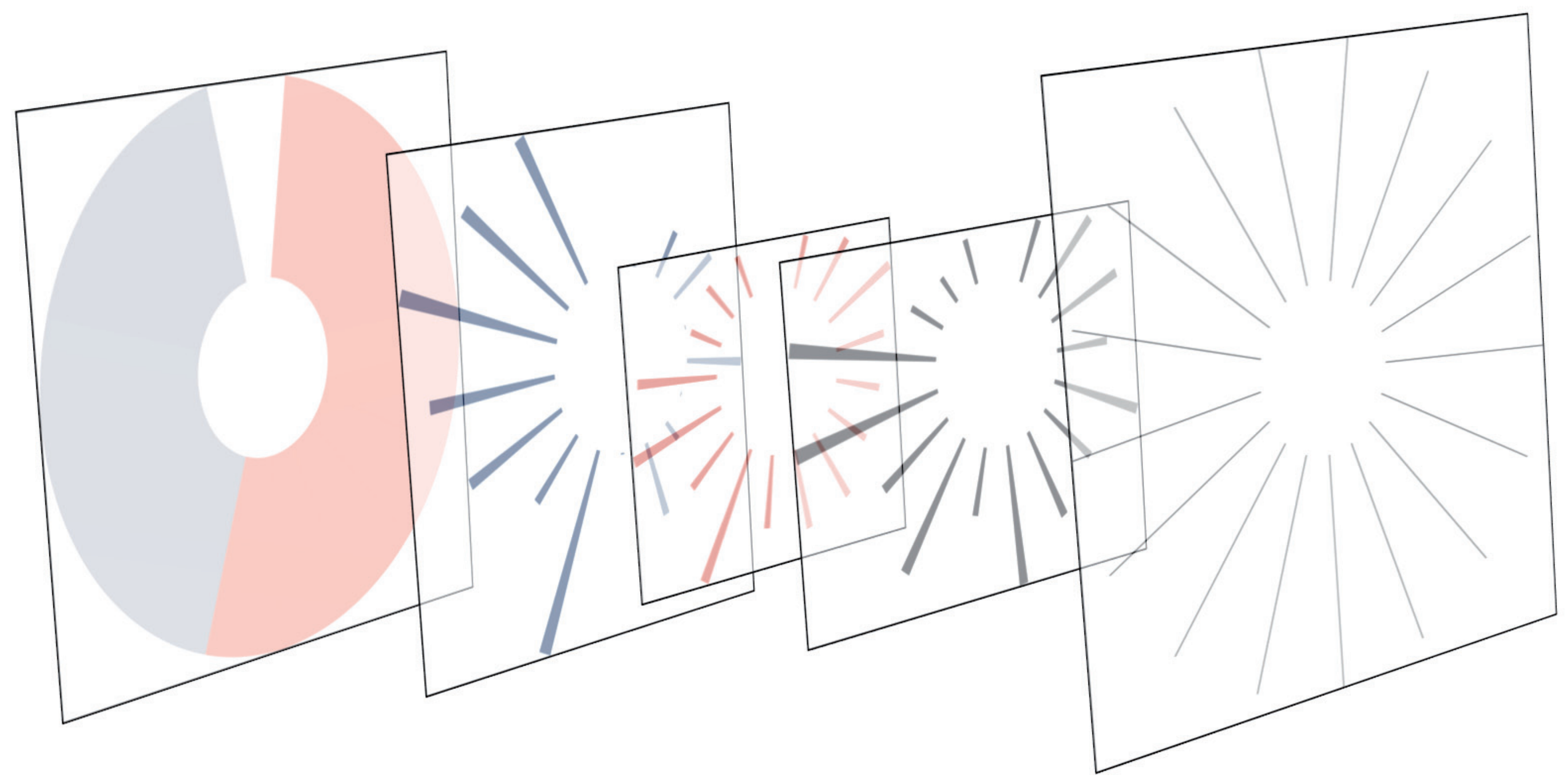
ProtoWiz presents a browser-based frontend to the powerful JavaScript visualization library **Protovis** (pun intended).

With **ProtoWiz**, we aim to enable users to build involved visualizations without having to touch JavaScript code while preserving as much of **Protovis'** flexibility as possible.

The various properties available in **Protovis** are represented as easy-to-use form fields. While the user interacts with the form, the visualization immediately re-renders itself, providing feedback about the effect of each property. The forms combine direct input with sensible pre-sets to make it as easy as possible to get a visualization working.


dots > root

dots
type: Dot
based on: root anchor:
data: demo
left: use data/value d.x
apply scale x
bottom: use data/value d.y
apply scale y
size: use data/value
apply scale
angle: use data/value
apply scale
stroke style: use data/value d.z
apply scale c
alpha
fill style: use data/value
apply scale
alpha
shape circle
visible ☑
title
line width



ProtoWiz uses the layer metaphor to lay out the marks in a visualization. Adding, deleting and reordering are supported and update the visualization dynamically.

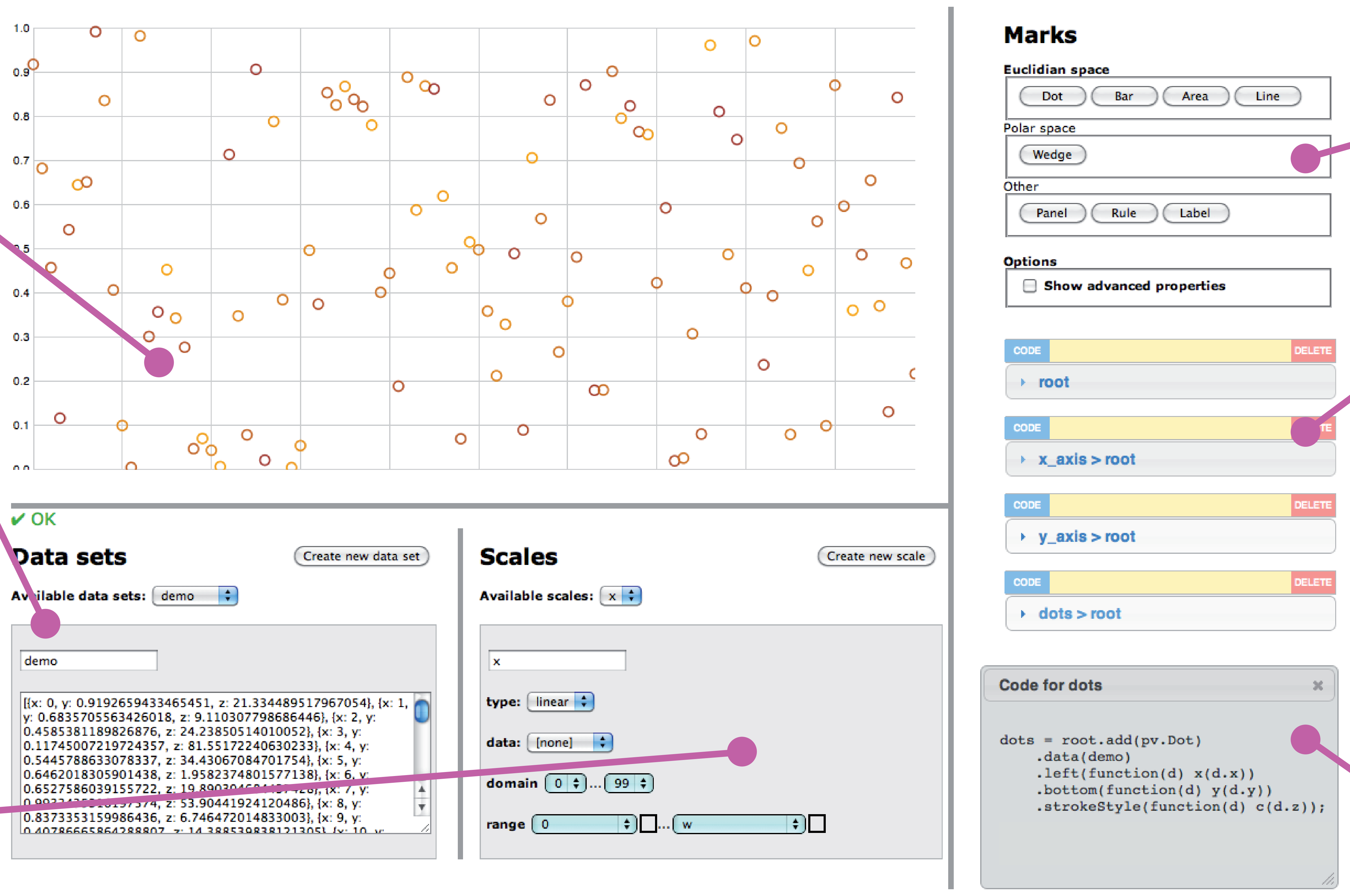
Supporting a graphical approach to visualization

Graphical decomposition is a key concept in **Protovis** — visualizations consist of separate layers containing general mark types. Their specific visual appearance is driven by the data associated with them. The layers metaphor is easily understood by users of graphics/design software as the same mental model applies there.

The visualization is generated on the fly based on the user's input.

Users can add their own data sets to work with. Currently, **ProtoWiz** recognizes JSON formatted data.

Scales are used to map values from the data domain to the visual domain. **ProtoWiz** supports numerical transforms and color scales.



Data sets

Available data sets: demo

demo

[{x: 0, y: 0.9192659433465451, z: 21.334489517967054}, {x: 1, y: 0.6835705563426018, z: 9.110307798686446}, {x: 2, y: 0.4585381189826876, z: 24.23850514010052}, {x: 3, y: 0.11745007219724357, z: 81.55172240630233}, {x: 4, y: 0.5445788633078337, z: 34.43067084701754}, {x: 5, y: 0.6462018305901438, z: 1.9582374801577138}, {x: 6, y: 0.6527586039155722, z: 19.89030[illegible]}, {x: 7, y: 0.99[illegible]4, z: 53.90441924120486}, {x: 8, y: 0.8373353159986436, z: 6.746472014833003}, {x: 9, y: 0.4078666586428880[illegible], z: 14.38853983812130[illegible]}, {x: 10, y:

Scales

Available scales: x

x
type: linear
data: [none]
domain 0 ... 99
range 0 ... w

Code for dots

```
dots = root.add(pv.Dot)
    .data(demo)
    .left(function(d) x(d.x))
    .bottom(function(d) y(d.y))
    .strokeStyle(function(d) c(d.z));
```

The basic mark types can be added with the click of a button.

The layers contain the marks' configuration and indicate inheritance relationships between marks.

Marks can show the **Protovis** code that generates them. This helps users understand **Protovis** programming.