VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE | DEPARTMENT OF COMPUTER SCIENCE

Data Visualization

460-4120

Fall 2024

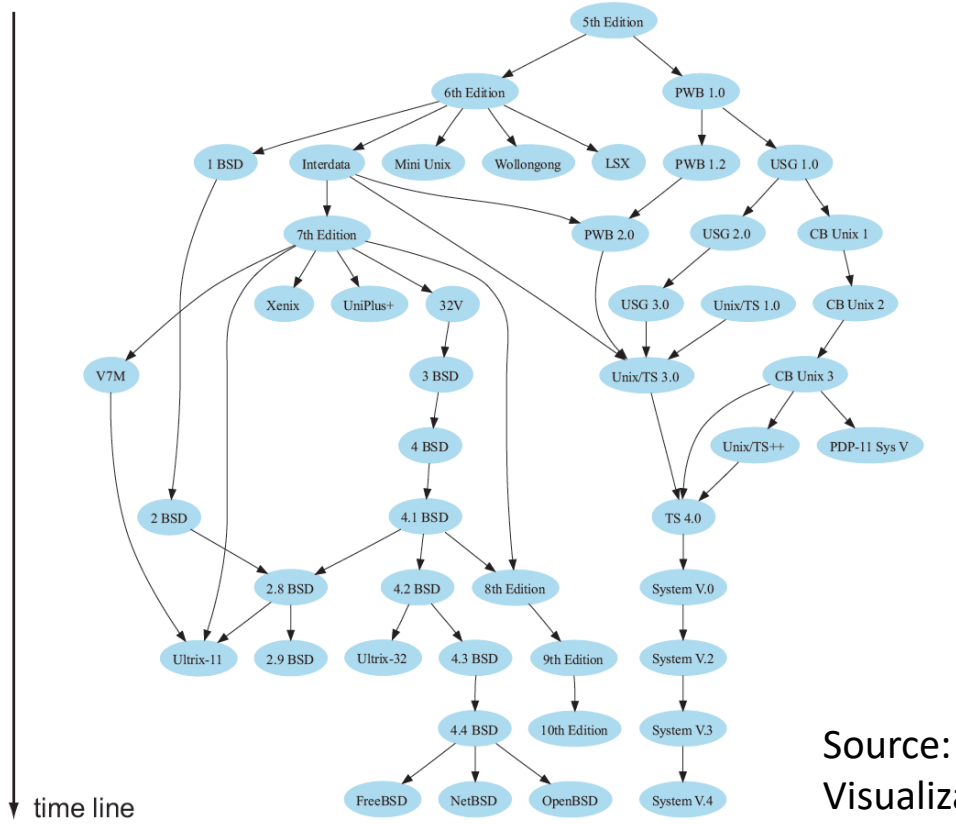
Last update 3. 12. 2024

Graph Visualization

- Hierarchical visualization – similar to node-link visualization, exploit the notion of hierarchy (graph may be naturally structured by means of nodes semantics which is inherent for trees)
- Two step algorithm
  - Nodes are assigned y-coordinates that are proportional to their layer numbers (nodes are grouped into layers where edges point from a node in a lower layer to a node in a higher layer)
  - Nodes in each layer (top-down) are permuted to minimize the number of edge intersections (expensive – use heuristics)
- Other methods: maximal layer width method and the depth-first search method

Graph Visualization

- Hierarchical graph layout example generated by GraphViz

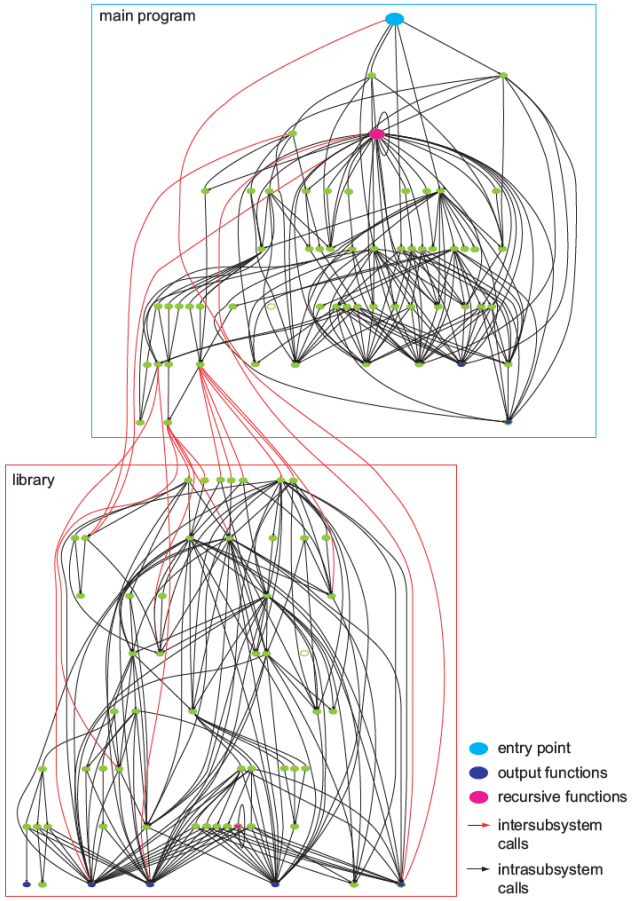


Source: Alexandru C. Telea, Data Visualization: Principles and Practice, 2014.

Graph Visualization

- More complex example

The call graph of a program visualized using a hierarchical graph layout. Call graphs are used in software engineering for understanding the structure of large software source code bases. Note the separation between the main program and library subsystem

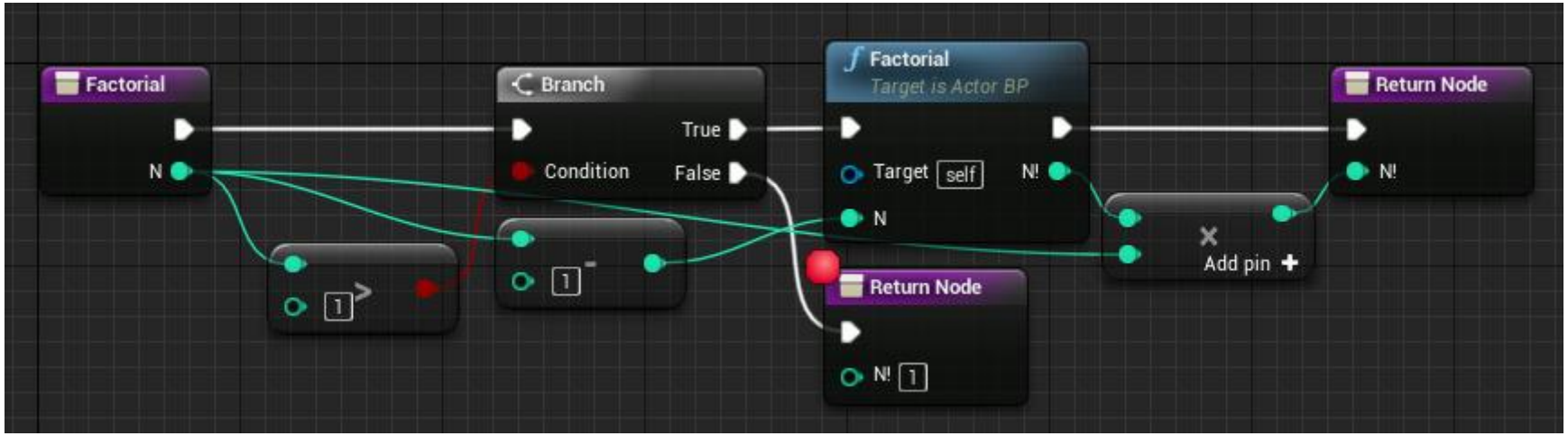


The edges, although carefully laid out using spline curves to minimize crossings, are still quite tangled and hard to tell apart from each other. Addressing this problem in general is quite difficult. We have two options:
1. Modify the graph to eliminate edges that are of little interest for the problem
2. Group related edges together, until a reduced edge count is reached
(we must know how to do it)

Source: Alexandru C. Telea, Data Visualization: Principles and Practice, 2014.

Graph Visualization

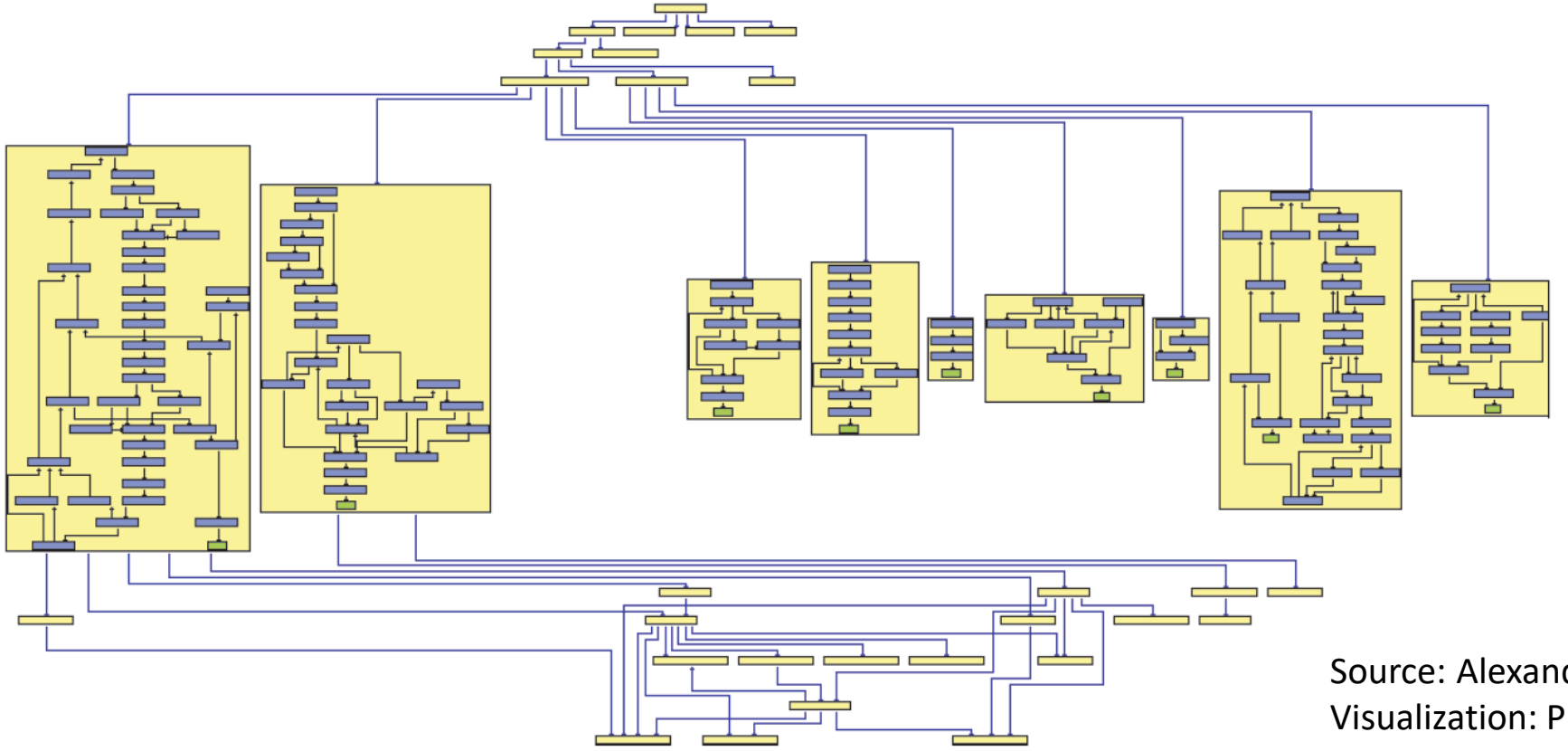
- Blueprints are visual scripting system used in Unreal Engine
- An example of encoding additional attributes in the graph visualization



Source: Unreal Engine Documentation.

Graph Visualization

- Hierarchical graph layout with orthogonal edge routing

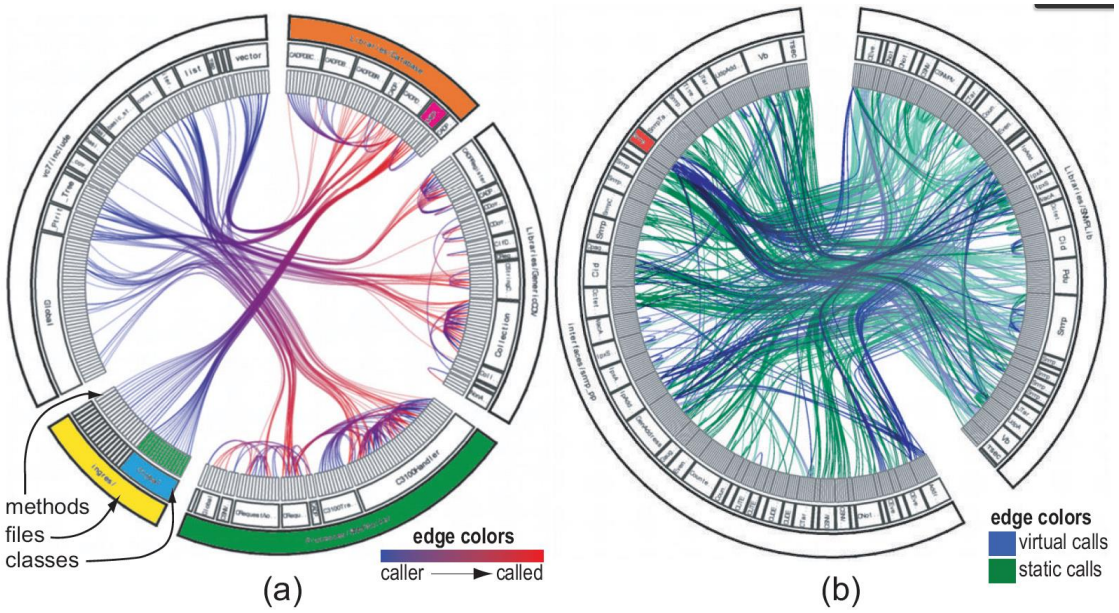


In contrast to the straight lines and splines, this orthogonal routing creates patterns that are arguably easier to follow. Note that different levels of detail are used throughout the layout

Source: Alexandru C. Telea, Data Visualization: Principles and Practice, 2014.

Graph Visualization

- Hierarchical edge bundling – a method for reduction of visual complexity when displaying large hierachical graphs
- Edges that are visually close to each other are visually grouped (bundled) to reduce clutter without lost of any information

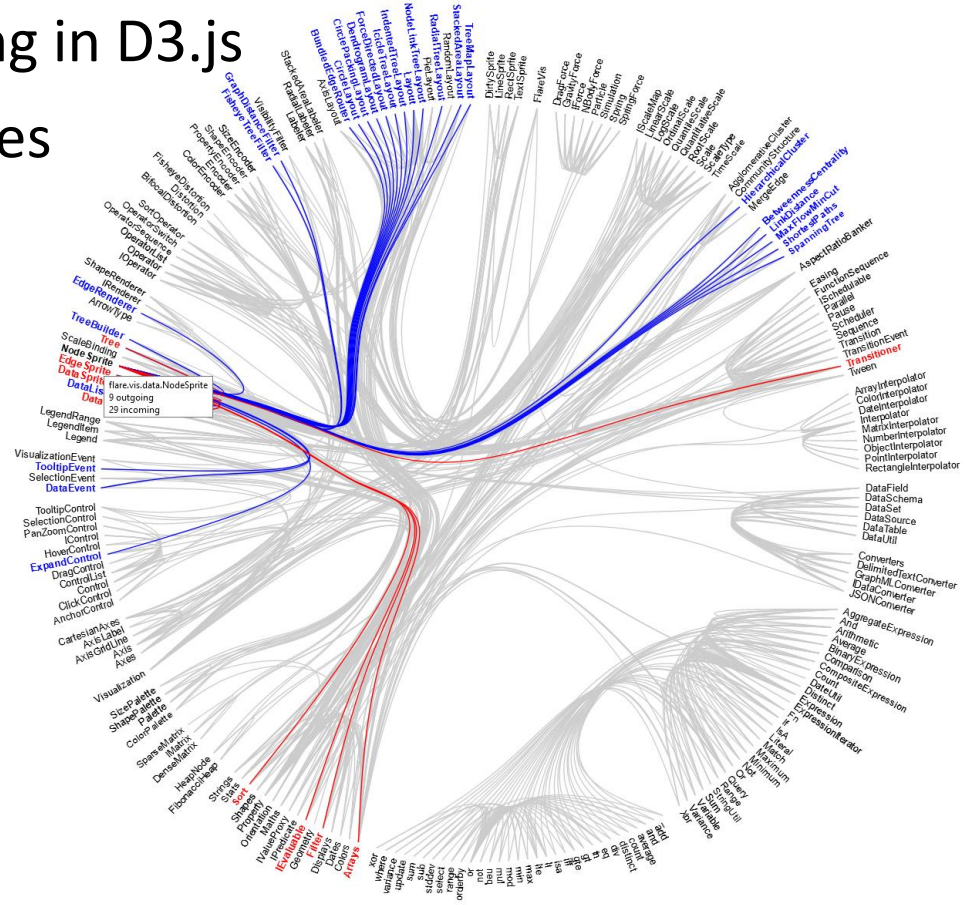


The layout used suggests that the left system is more modular than the right system.

Source: Alexandru C. Telea, Data Visualization: Principles and Practice, 2014.

Graph Visualization

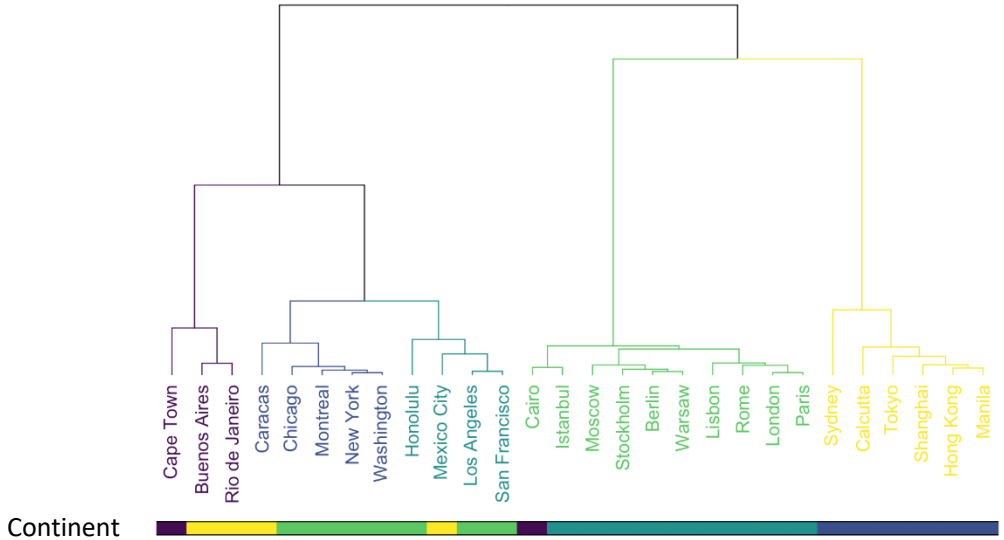
- Interactive hierarchical edge bundling in D3.js
- Radial dendograms representing trees



Source: https://observablehq.com/@d3/hierarchical-edge-bundling

Graph Visualization

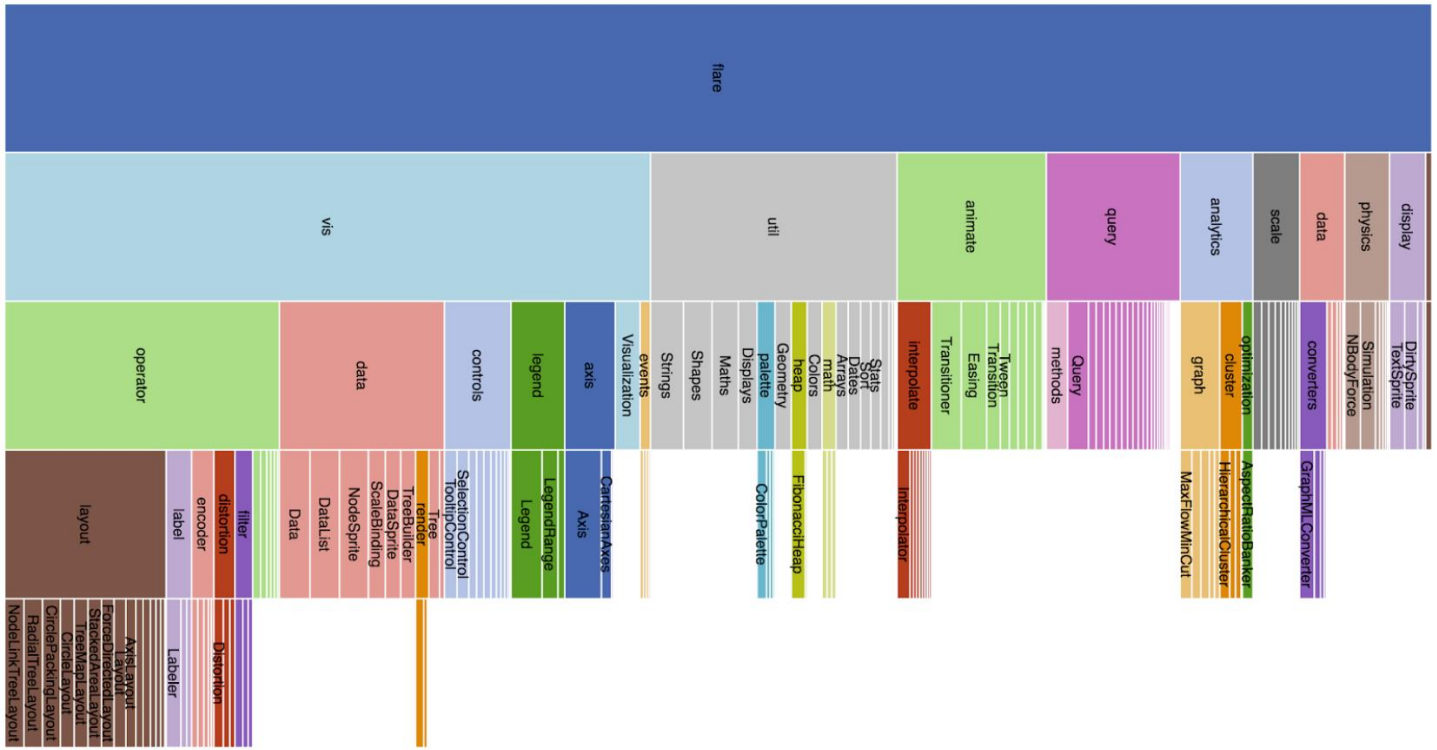
- This graphic allows to validate that the clustering indeed grouped cities by continent. There are a few discrepancies that are logical. Indeed, Mexico city has been considered as a city of South America here, although it is probably closer from North America as suggested by the clustering.



Source: https://www.data-to-viz.com/graph/dendrogram.html

Graph Visualization

- Icicle plots



Icicle plots are a method for presenting hierarchical/clustered data. The technique was developed in 1983 by Kruskal and Landwher. They were named as such due to the fact that the clustering in the visualization looks like icicles.

Source: https://www.cs.middlebury.edu/~candrews/showcase/infovis_techniques_s16/icicle_plots/icicleplots.html

Graph Visualization

- Ascii style (old school) icicle plot example (no filtering, no zoom, no shading, no interactivity, no exploration)

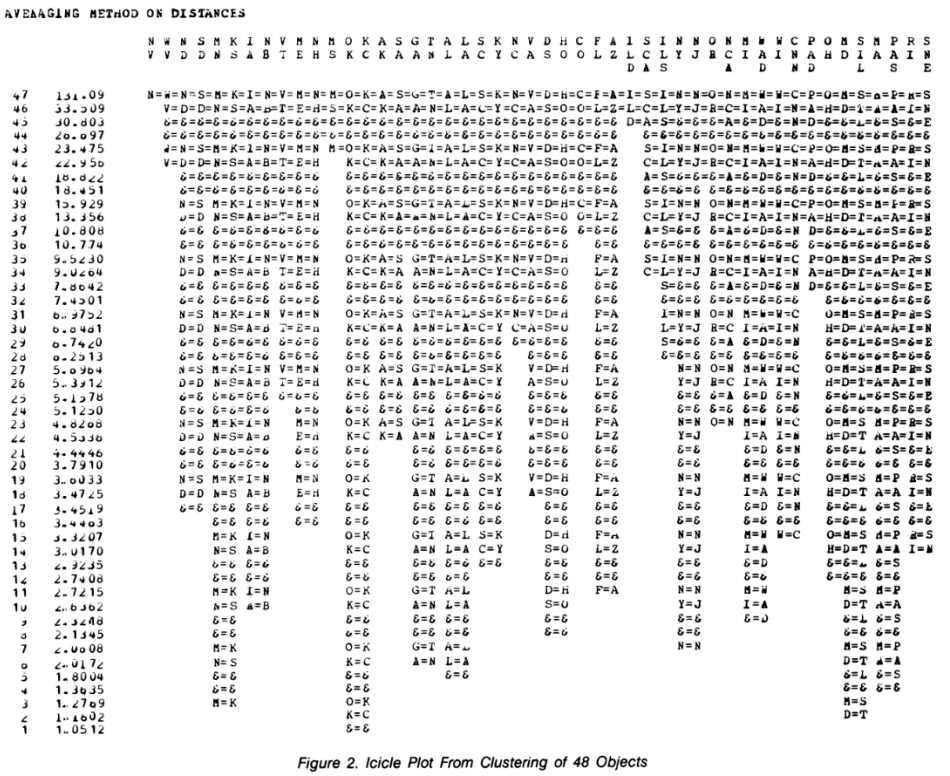


Figure 2. Icicle Plot From Clustering of 48 Objects

Source: https://www.cs.middlebury.edu/~candrews/showcase/infovis_techniques_s16/icicle_plots/icicleplots.html

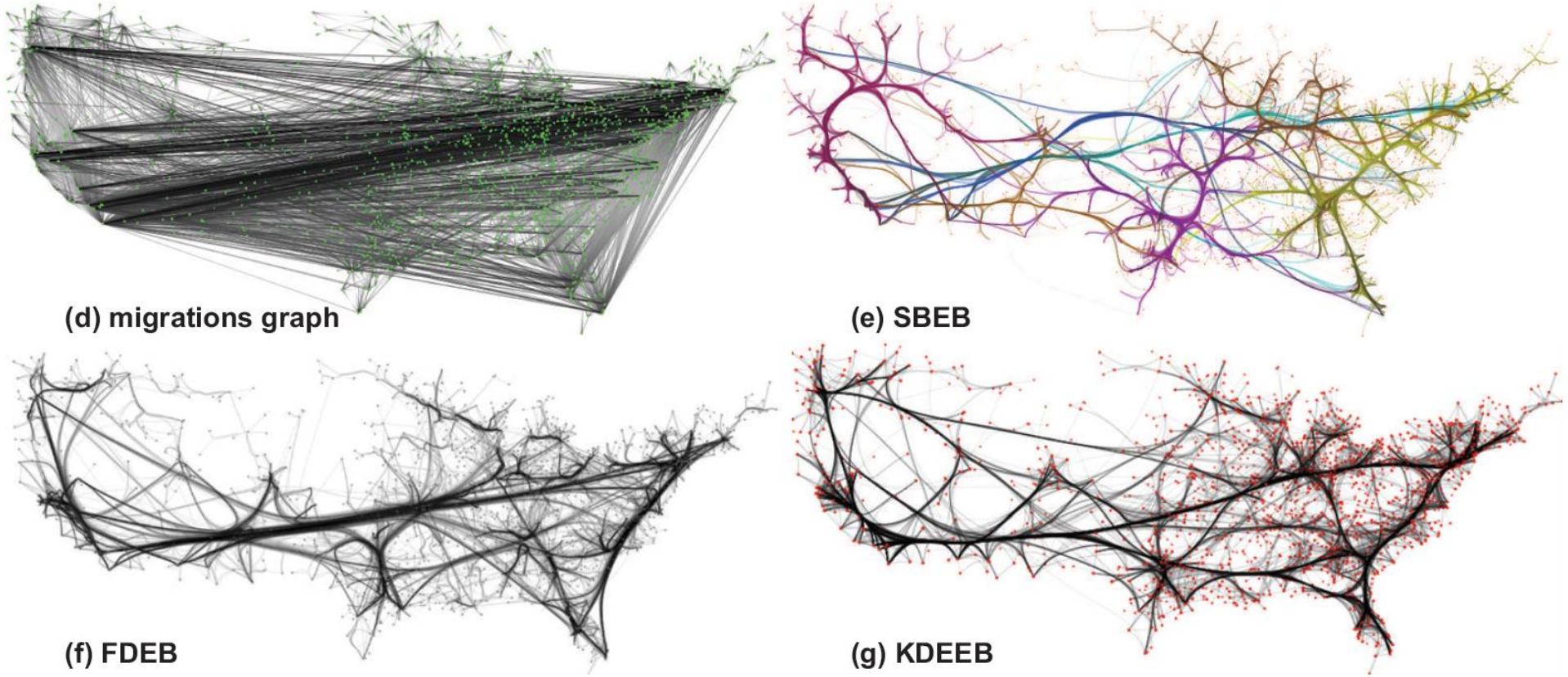
Graph Visualization

- General graph-edge bundling methods
  - Force-directed edge bundling (FDEB)
  - Geometry-based edge bundling (GBEB)
  - Winding roads (WR)
  - Skeleton-based edge bundling (SBEB)
  - Kernel density estimation edge bundling (KDEEB)

Graph Visualization

Source: Alexandru C. Telea, Data Visualization: Principles and Practice, 2014.

- Examples of general graph-edge bundling methods



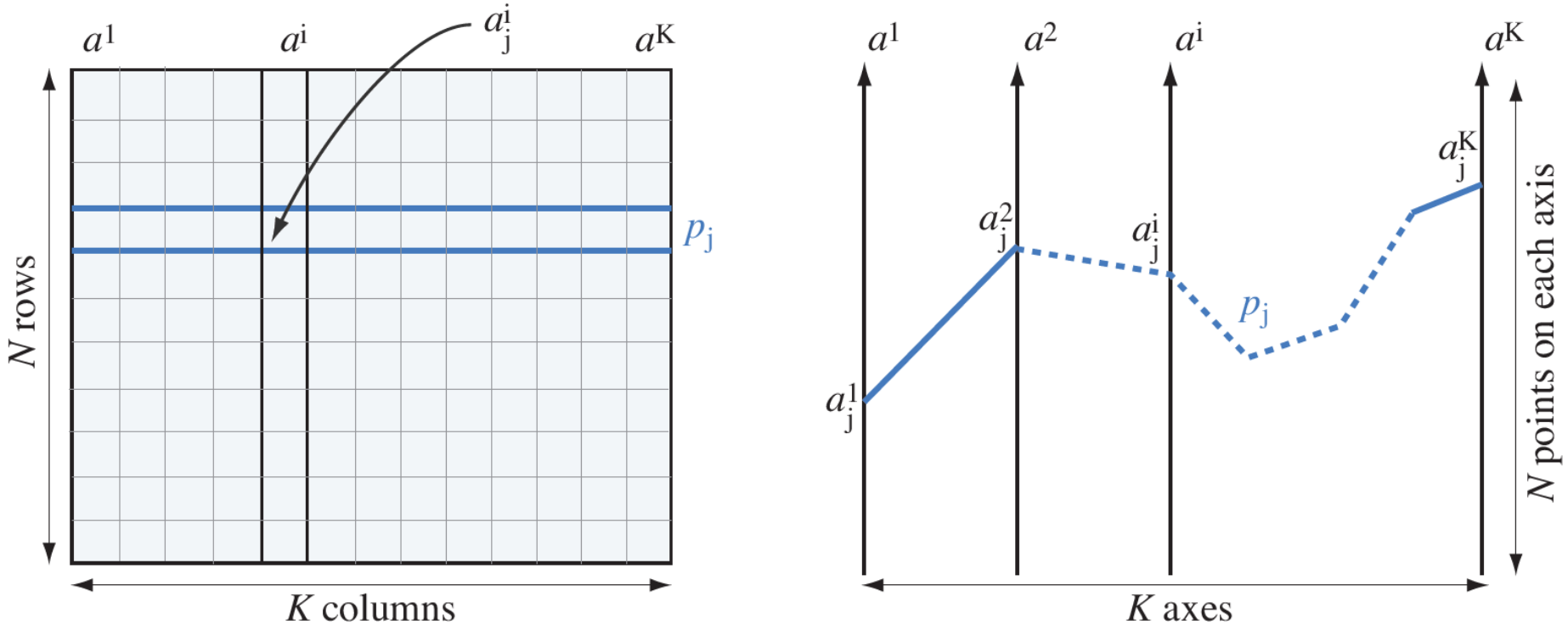
Multivariate Data Visualization

- Consider a set of N data points $D = \{\boldsymbol{p}_i\}$, $1 \leq i \leq N$
- Every data point $\boldsymbol{p}_i$ is represented as a K-dimensional vector of attributes $\boldsymbol{p}_i = \left(a_i^1, \ldots, a_i^K\right) \in A^K$ where A is some domain
- Dataset $\{\boldsymbol{p}_i\}$ is called multivariate
- We want to visualize this dataset such that correlations, outliers, clusters, and trend become visible

Multivariate Data Visualization

Source: Alexandru C. Telea, Data Visualization: Principles and Practice, 2014.

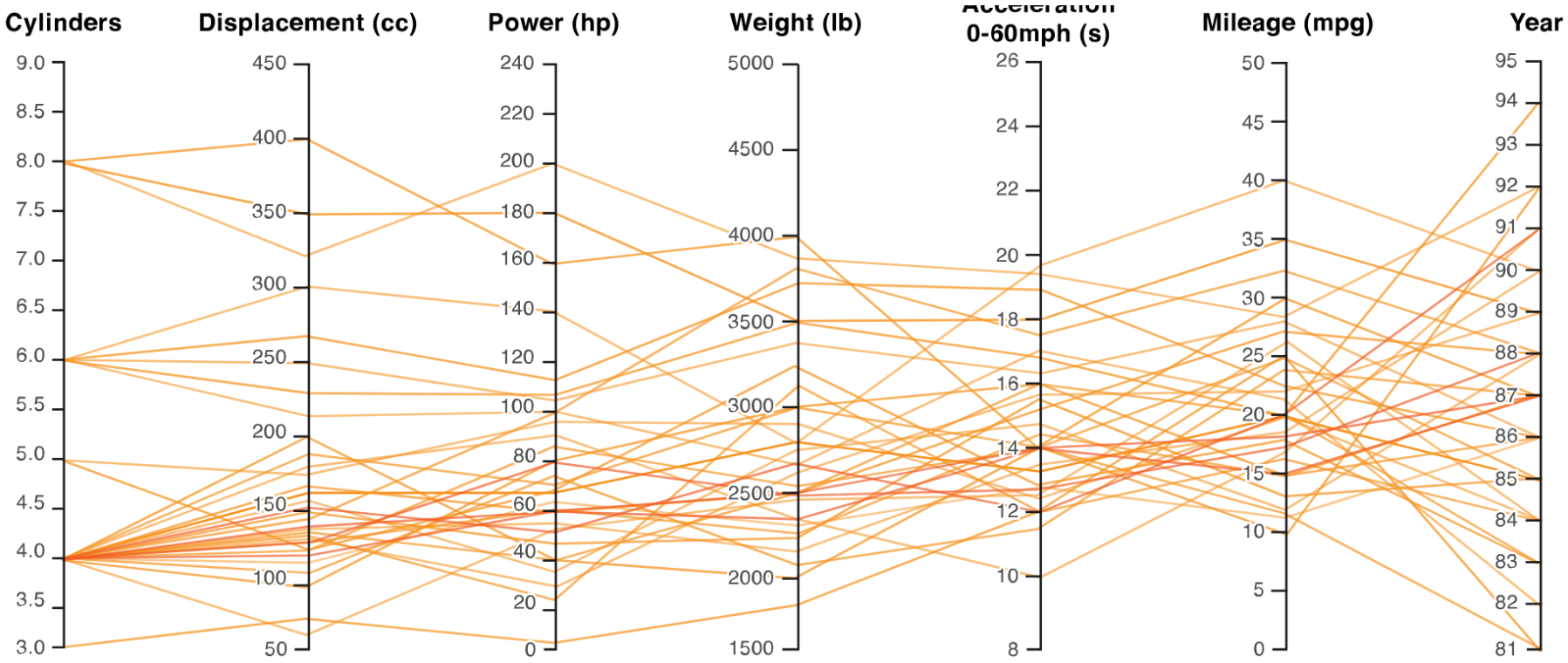
- Example of table visualization (left) and parallel coordinate plot (PCP) of K-dimensional point p_j (right)



Multivariate Data Visualization

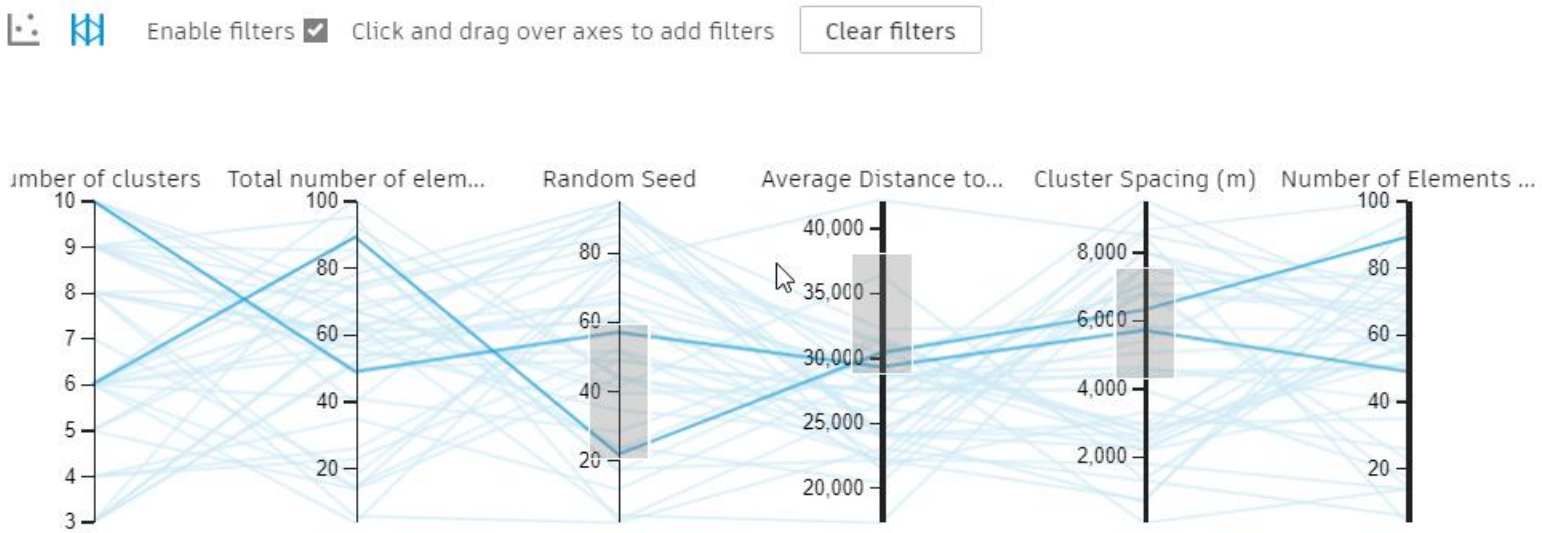
Source: https://datavizcatalogue.com/methods/parallel_coordinates.html

- Parallel coordinate plot showing 7 attributes for about 30 cars



Multivariate Data Visualization

- The graph can be (interactively) filtered by dragging the selection on each vertical axis



Source: https://www.generativedesign.org

Dimensionality Reduction

- Consider the same multivariate dataset $D \subset A^K$ as before, i.e. $D = \{\boldsymbol{p}_i\}$ where $\boldsymbol{p}_i$ lives in some K-dimensional space A^K
- We want to visualize the structure of the dataset D
- To do this, we construct so-called projection function

$$P\colon A^K \to \mathbb{R}^k$$

where k is typically 2 or 3 what yields 2D or 3D scatter plot (graph splatting)

- Projection function P should respect several constraints
  - Distance preservation
  - Neighborhood preservation

Dimensionality Reduction

- Stress function – global indicator of constraints preservation

$$\sigma = \sqrt{\frac{\sum_{i,j}\left(\left\|\boldsymbol{p}_i - \boldsymbol{p}_j\right\| - \left\|P(\boldsymbol{p}_i) - P\left(\boldsymbol{p}_j\right)\right\|\right)^2}{\sum_{i,j}\left\|\boldsymbol{p}_i - \boldsymbol{p}_j\right\|^2}}$$

- This function measures how well the placement of the projections preserves the aforementioned constraints
- Techniques that compute a projection P that minimizes stress function σ are known as dimensionality reduction methods

Dimensionality Reduction Techniques

- Multidimensional scaling
- Projection-based dimensionality reduction

Multidimensional Scaling

- Instead of actual coordinates of the points $\boldsymbol{p}_i$ in A^K we only know the square matrix $M_{N\times N} = \{d_{i,j}\}$ where $1 < i < N$, $1 < j < N$ and $d_{i,j}$ are the distances (or dissimilarities) between these K-dimensional points
- Distances are computed with the aim of arbitrarily designed function $\delta: A \times A \rightarrow \mathbb{R}^+$ which gives the one-dimensional distance between two attributes such that

$$d_{i,j} = \|\boldsymbol{p}_i - \boldsymbol{p}_j\| = \sqrt{\sum_{l=1}^{K} \delta\left(a_i^l, a_j^l\right)^2}$$

- Multidimensional scaling (MDS) is the group of methods that compute the projection P by directly minimizing the stress function σ

Multidimensional Scaling

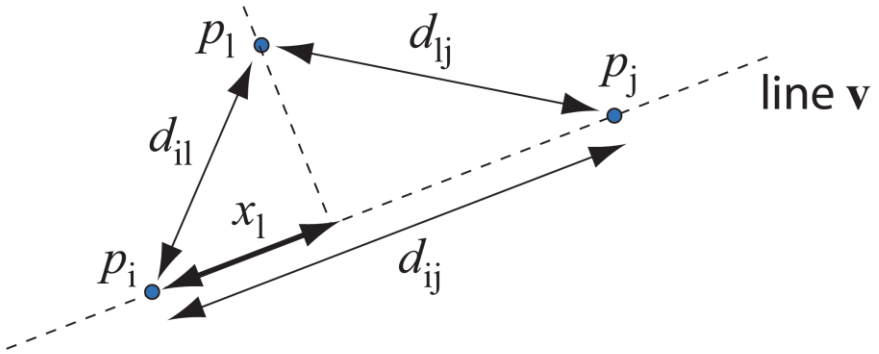
- Embeding – process of assigning k-dimensional coordinates to points in an unknown K-dimensional space
- Word scaled means that the distances between data points in the low-dimension k should be scaled distances between the same data points in the original K-dimensional (unknown) space
- **Force-directed layouts** – the edge stiffness between two points is inversely proportional to their distance (it has high complexity $O(N^2)$; optimization: distant points are not connected by an edge at all)
- **FastMap** – uses only the distance matrix M
  - 1. Choose points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ which maximize $d_{i,j}$
  - 2. Project all points $\boldsymbol{p}_l$ on the line $\boldsymbol{v} = \boldsymbol{p}_j - \boldsymbol{p}_i$ to find coordinate in the k-dimensional space
  - 3. Recursively apply FastMap to the projections of $\boldsymbol{p}_i$ on a plane orthogonal to $\boldsymbol{v}$, to find the remaining $k - 1$ coordinates

Multidimensional Scaling

- To find the coordinate x_l of $\boldsymbol{p}_l$ along the line $\boldsymbol{v}$, we only need to know the distances between the points $\boldsymbol{p}_i$, $\boldsymbol{p}_j$, and $\boldsymbol{p}_l$ using the cosine law theorem

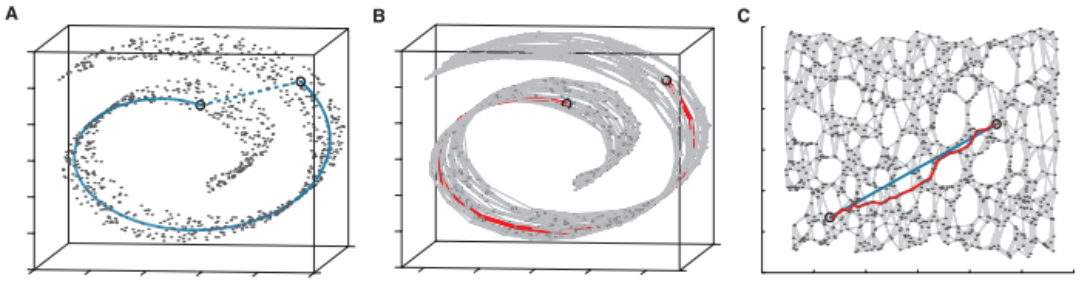
$$x_l = \frac{d_{i,l}^2 + d_{i,j}^2 - d_{l,j}^2}{2d_{i,j}}$$

- FastMap has complexity only $O(kN)$



Multidimensional Scaling

- Other methods
  - Spectral decomposition – project points along the eigenvectors having the largest eigenvalues of the distance matrix
  - LLE – topology preserving manifold learning method
  - Isomap – captures nonlinear relationships in the dataset where point-to-point distances are replaced by an approximation of the geodesic distance between points given by the shortest path on a graph created connecting neighbor points in the K-dimensional space with the original distance as weight



Source: http://benalexkeen.com/wp-content/uploads/2017/05/isomap.png

Projection-Based Dimensionality Reduction

- Applicable when we know the original K-dimensional point coordinates
- Karhunen-Loève method (K-L transform) works as follows
  - 1. Compute covariance matrix C of the N K-dimensional points $\boldsymbol{p}_i$
  - 2. Compute the eigenvectors $\boldsymbol{e}_i$ of C corresponding to the first k largest eigenvalues λ_i of C
  - 3. Compute the projections $P(\boldsymbol{p}_i) = \left(q_i^1, \ldots, q_i^k\right)$ as $q_i^l = \boldsymbol{e}_l \cdot \boldsymbol{p}_i$ for all $1 < l < k$
- Idea behind: eigenvectors corresponding to the largest eigenvalues of C indicate the direction in K-dimensional space along which the points $\boldsymbol{p}_i$ spread the most. If we construct our k-dimensional projections $P(\boldsymbol{p}_i)$ by projecting data along these directions, we preserve the most information encoded in interpoint distances
- It is closely related to the singular value decomposition (SVD) technique

Advanced Dimensionality Reduction Techniques

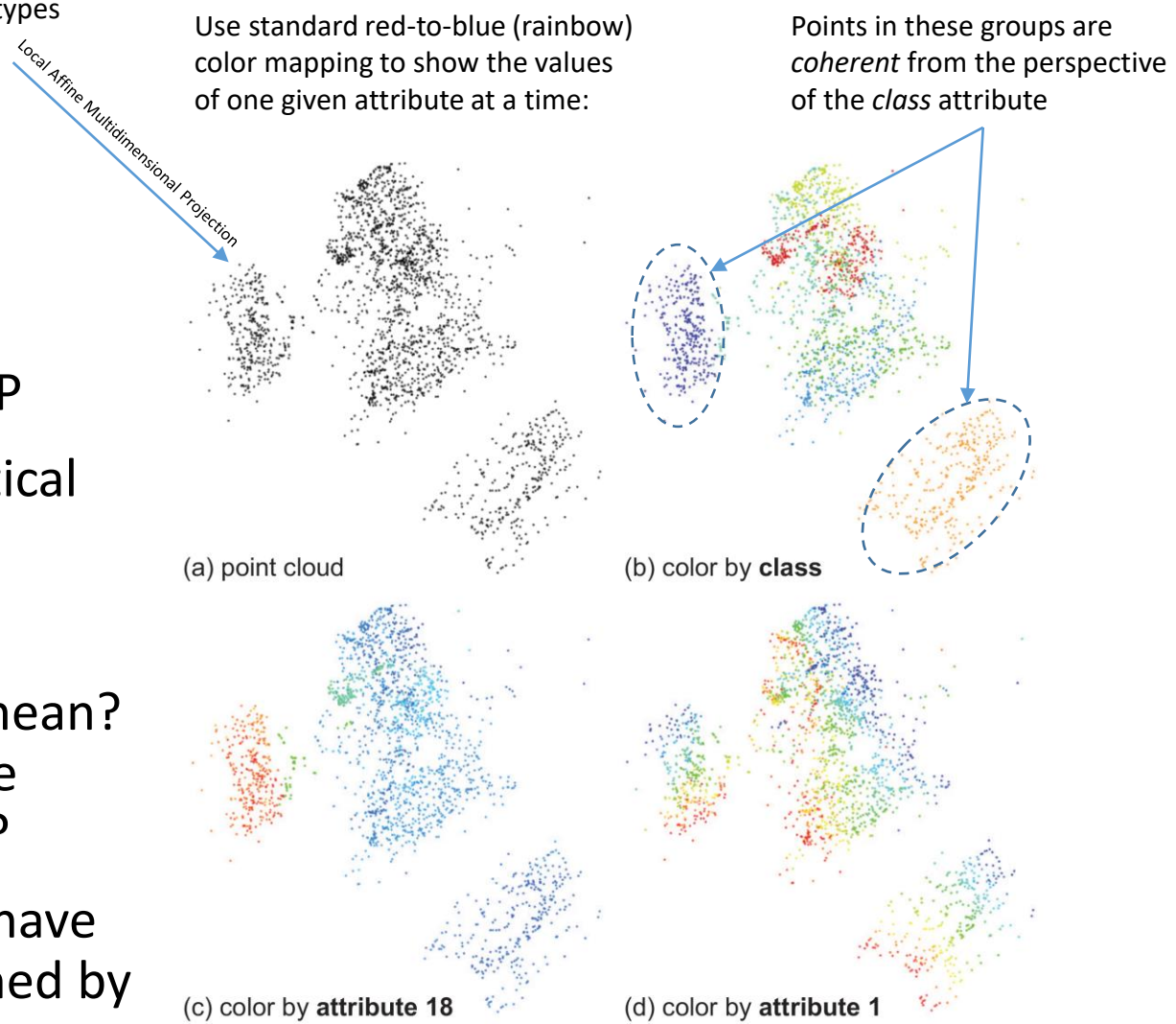
- Optimize balance between scalability (handle large datasets) and accuracy (preserve distances and neighborhoods)
- Least square projection (LSP) – very precise in preserving neighborhoods
- Part-linear multidimensional projection (PLMP)
- Local affine multidimensional projection (LAMP) – needs to access the point coordinates, very fast

Segmentation dataset: Each point describes a small image block randomly chosen from 7 natural outdoor images of different types

Projections Examples

- 2D scatter plot of 2100 points from 18-dimensional dataset projected in 2D by LAMP
- Each of 18 attributes represents some statistical properties of observed objects
- One additional attribute describes the class
- How to read the plot? What do the groups mean? What combination of attributes and attribute values is typical for points in a group (if any)?
- Do the projections and the resulting groups have precise meaning, or are they partly determined by the limitations of the projection algorithm?

Source: Alexandru C. Telea, Data Visualization: Principles and Practice, 2014.



Attribute 18, only the left group have high (red) values for this attribute. **Hence, the left group, and its class value, can be explained in terms of high-values of attribute 18**

Attribute 1, albeit reflected in the positions of the projected points, does not explain the perceived point groups

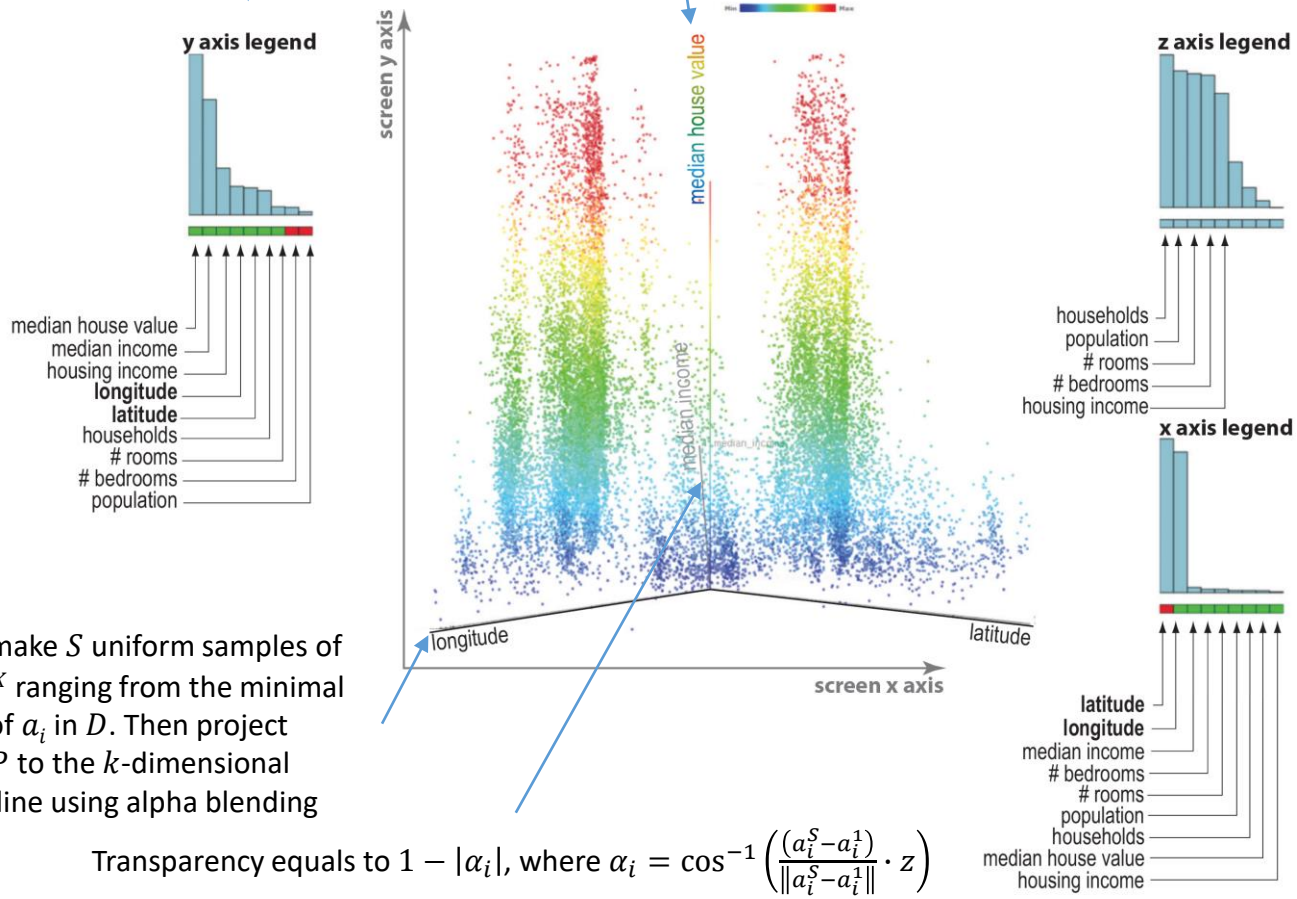
Household dataset: Each of 206490 9-dim points represents a household parameters recorded by the California census of 1990

Projection Explanations

- Explanatory visualization mechanisms annotate 2D or 3D projection plot with information that enables users to revert the k-dimensional mapping to the original K-dimensions. Here, the k is 3
- The attribute axes convey six important insights that help us understand the meaning of the scatter plot (orientation, visibility, length/visible variance, angles/correlation, direction, curvatures/linearity of P)

Legends show the *mix* of original attributes that each screen axis depicts (scaled based on the $|\alpha_i|$ between the projected axis and the screen axis)

Color a user-chosen axis with the values of its attribute at the sample points



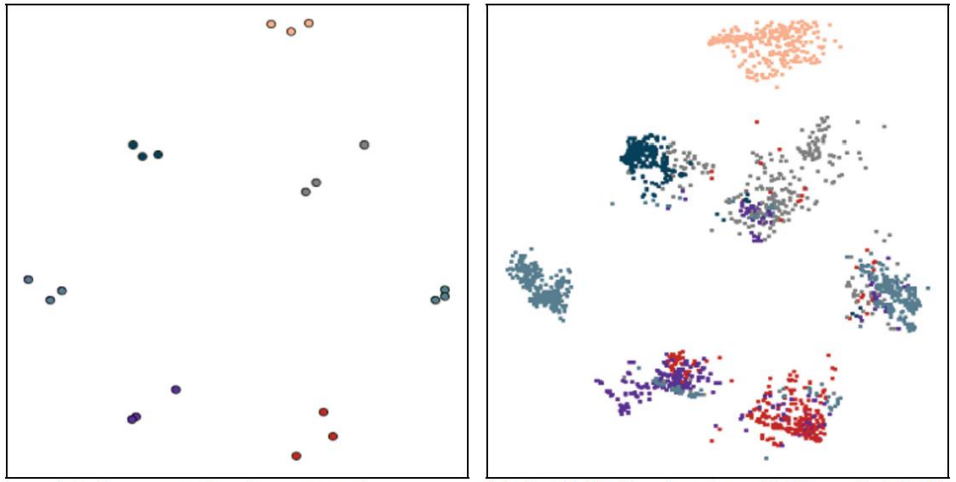
For each attribute a_i, make S uniform samples of the line $a_i^1, \ldots, a_i^S$ in A^K ranging from the minimal to the maximal value of a_i in D. Then project sampled points using P to the k-dimensional space and draw a polyline using alpha blending

Transparency equals to $1 - |\alpha_i|$, where $\alpha_i = \cos^{-1}\left(\frac{(a_i^S - a_i^1)}{\|a_i^S - a_i^1\|} \cdot z\right)$

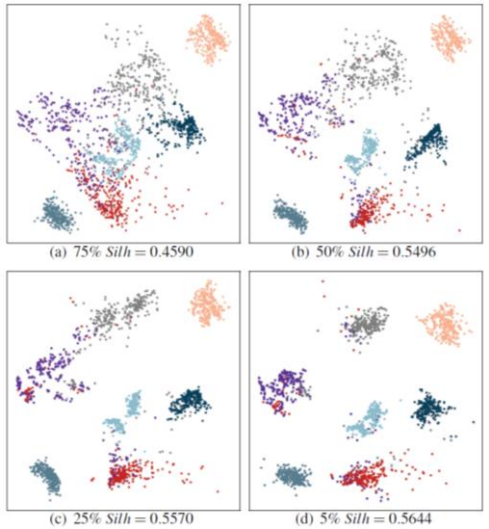
Source: Alexandru C. Telea, Data Visualization: Principles and Practice, 2014.

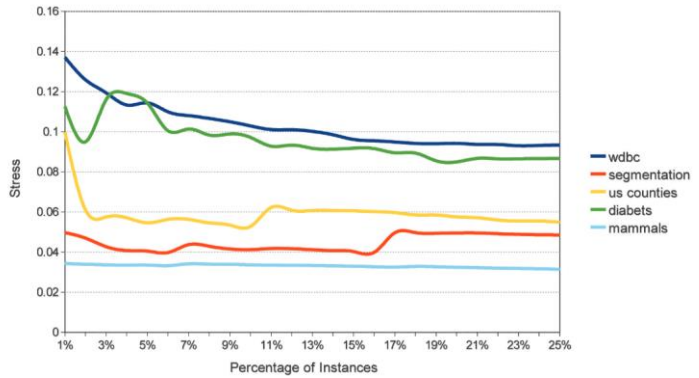
Local Affine Multidimensional Projection (LAMP)

- Works directly with the point coordinates, interactive, very fast, precise, support visualization-based data correlation
- To allow more user control over the final layout, LAMP provides user-controlled mapping of selected points from the dataset
- Has restrictions regarding the number of dimensions against the number of points



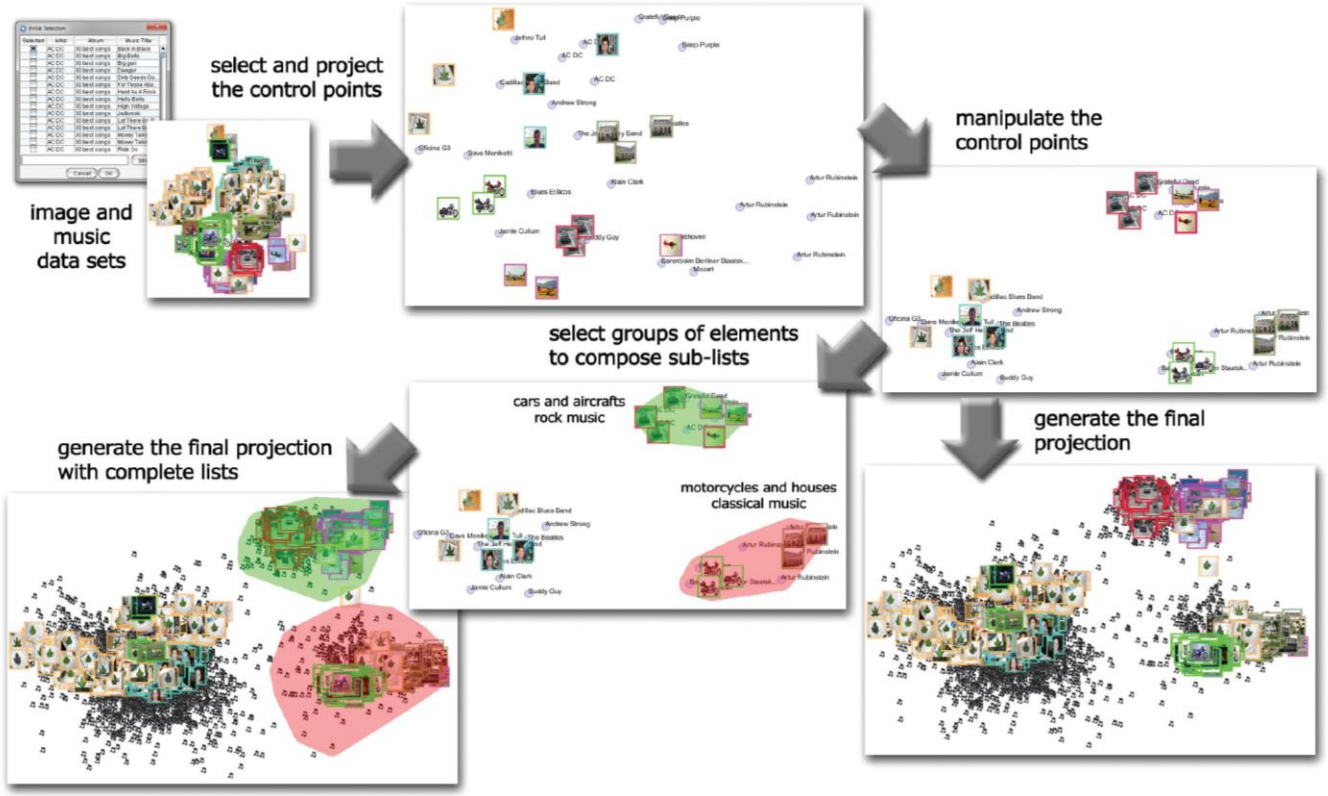
(a) 3 control points per class. (b) LAMP Projection ($Silh = 0.4361$)





Source: JOIA, Paulo, et al. Local affine multidimensional projection, 2011. https://github.com/lgnonato/LAMP

Local Affine Multidimensional Projection (LAMP)

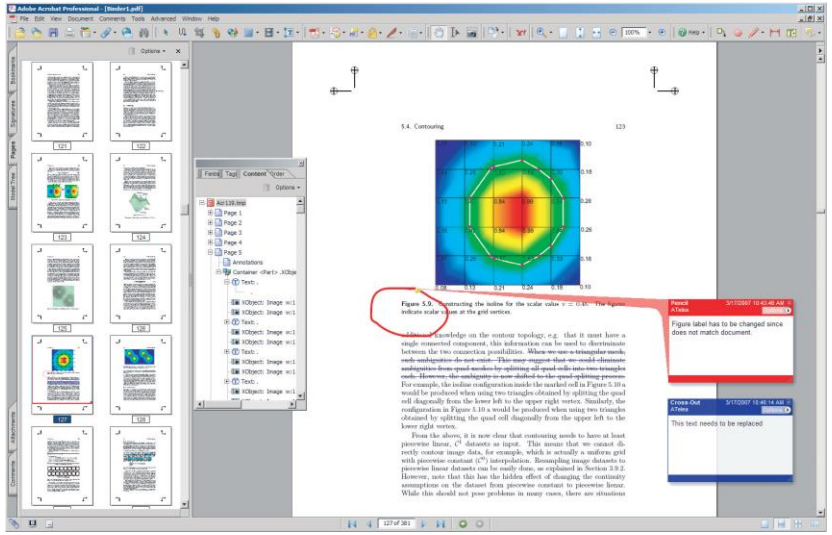


A few representatives (control points) of music and pictures are selected from the corresponding data sets (top left) and placed in the visual space (top middle). User interacts with the sample pictures and music so as to bring close the ones to be correlated (top right). LAMP maps music and picture according to the user provided grouping (bottom right). The user can brush multiple regions in the visual space (bottom middle), each one corresponding to the pictures and music the will make up a slide show (bottom left).

Source: JOIA, Paulo, et al. Local affine multidimensional projection, 2011.

Text Visualization

- Three categories of information: content (actual text), structure (paragraphs, sections, chapters, or other elements), metadata (information related to the text that are not contained in the text itself – information about the document itself, cross references, keywords, indexes, authors, publisher, publication date)
- Content-based visualization



(a) The document's detailed content
(b) A page-level overview
(c) The document structure
(d) Annotation metadata

Source: Alexandru C. Telea, Data Visualization: Principles and Practice, 2014.

Exercise

- Try to apply the FastMap algorithm on dimensionality reduction of some multidimensional dataset (where $K > 4$ and $k = 2$)
- http://www.cs.cmu.edu/~christos/software.html
- The result should be a 2D colored scatter plot with highlighted ranges of selected attributes or actual classes of the original data points
- You may use some trivial K-dim dataset rotated with a (random) K-dim rotation matrix obtained from mgen library (https://github.com/NOhs/mgen)
- For further reference, see the original publication: FALOUTSOS, Christos; LIN, King-Ip. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In: *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*. 1995. p. 163-174.

Tools for Data Visualization

- **Plotly** (https://github.com/plotly/plotly.js, open source library)
- High-level, declarative charting library in JavaScript built on top of d3.js and stack.gl
- Supports over 20 chart types, including scientific charts, 3D graphs, statistical charts, SVG maps, financial charts etc.
- Useful for building data visualization apps with highly custom user interfaces
- Version for Python (plotly.py) and R also exists

Tools for Data Visualization

- **Dash** (https://plot.ly/dash, open source library)
- Python framework for building web applications (rendered in the web browser) written on top of Flask, Plotly.js, and React.js
- Useful for building data visualization apps with highly custom user interfaces

Tools for Data Visualization

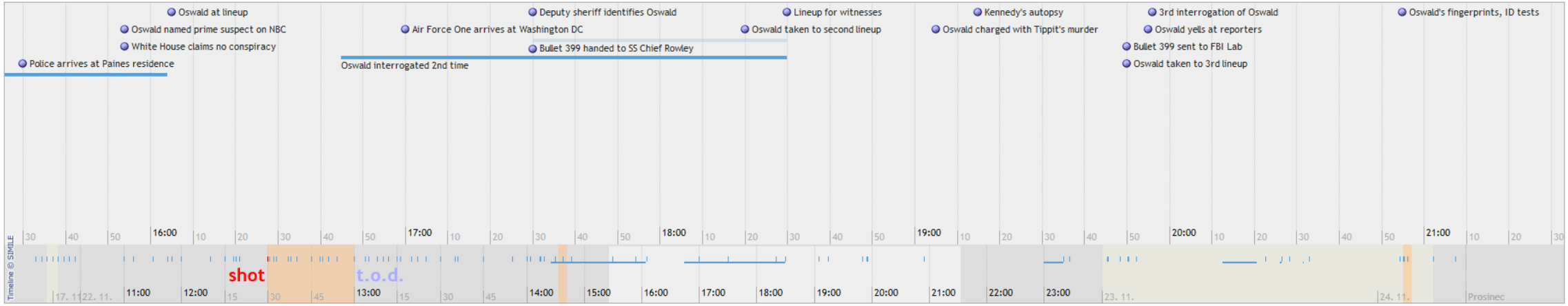
- **Chart.js** (http://www.chartjs.org/, available under the MIT license)
- Simple HTML5 Charts using the canvas element
- Useful for building data visualization apps with highly custom user interfaces

Tools for Data Visualization

- **Google Charts** (https://developers.google.com/chart/interactive/docs, available under the Apache license)
- Simple HTML5 Charts using the canvas element
- Useful for building data visualization apps with highly custom user interfaces

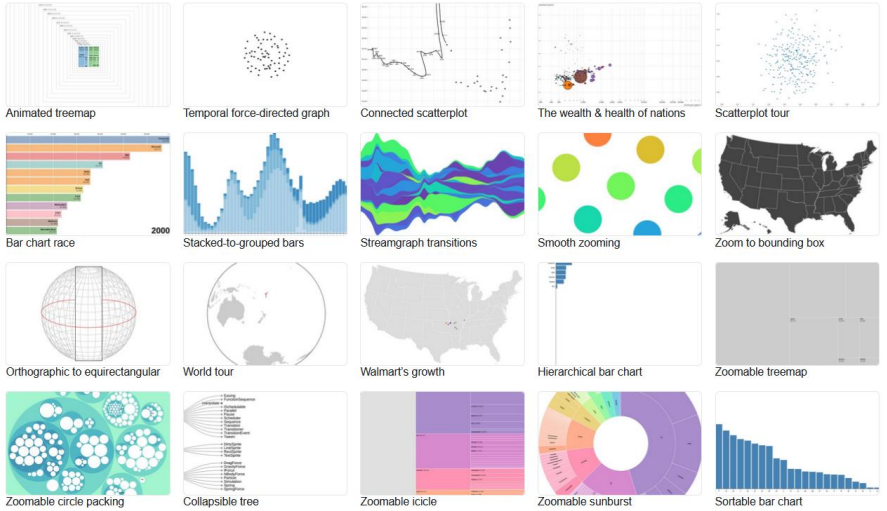
Tools for Data Visualization

- **Timeline** (http://www.simile-widgets.org/timeline, available under the BSD license)
- Web widget for visualizing temporal fata in form of interactive timelines



Tool: D3 = Data Driven Documents

- What is D3
  - A JavaScript library for manipulating documents based on data
  - Visualizing Data with common Web standards like HTML, CSS, SVG
  - Constructing the DOM from Data
  - Each data point has a corresponding element
- What D3 is not?
  - Not a chart library; it is a visualization library
  - Not a compatibility layer
  - Not only about SVG, HTML, or Canvas
- See https://d3js.org for further reference



D3: Selection

- Modifying documents using W3C DOM API is tedious:

- ```
  var paragraphs = document.getElementsByTagName("p");
  for (var i = 0; i < paragraphs.length; i++)
  {
      var paragraph = paragraphs.item(i);
      paragraph.style.setProperty("color", "white",    null);
  }
  ```
  ```

D3: Selection

- D3 employs a declarative approach:

- Operating on arbitrary sets of nodes:
- `d3.selectAll("p").style("color", "white");`

- Manipulating individual nodes:
- `d3.select("body").style("background-color", "black");`

D3: Selection

- D3 uses CSS Selectors
- Single selector
  - `#foo` `// <any id="foo"> </any>`
  - `foo` `// <foo> </foo>`
  - `.foo` `// <any class="foo"> </any>`
  - `[foo=bar]` `// <any foo="bar"> </any>`
  - `foo bar` `// <foo><bar> </bar></foo>`
- Multiple selectors:
  - `foo.bar` `// <foo class="bar"> </foo>`
  - `foo#bar` `// <foo id="bar"> </foo>`

D3: Select and Modifiy Element Properties

- ```
  var svg = d3.select("svg");
  ```

- ```
  var rect = svg.select("rect");
  rect.attr("width", 100);
  rect.attr("height", 100);
  rect.style("fill", "steelblue");
  ```

- ```
  svg.select("rect")
  .attr("width", 100)
  .attr("height", 100)
  .style("fill", "steelblue");
  ```

- ```
  d3.select("svg").select("rect")
  .attr({
  "width": 100,
  "height": 100
  })
  .style({
  "fill": "steelblue"
  });
  ```

D3: Transitions

```
var svg = d3.select("svg");

  svg.selectAll("rect")
  .data([127, 61, 256])
  .transition()
  .duration(1500) // 1.5 second
  .attr("x", 0)
  .attr("y", function(d,i) { return i*90+50; })
  .attr("width", function(d,i) { return d; })
  .attr("height", 20)
  .style("fill", "steelblue");
```

D3: Setup

- Create a new folder for your project
- Within that folder create a subfolder called d3
- Download the latest version of D3 into that subfolder and decompress the ZIP file
- (notice both the minified and standard version)
- Or, download entire repository:

https://github.com/mbostock/d3

- Or, to link directly to the latest release, copy this snippet:

```
<script src="//d3js.org/d3.v3.min.js" charset="utf-
8"></script>
```

D3: Setup

- Create a simple HTML page within project folder named index.html:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html lang="en">
  <head>
  <meta http-equiv="content-type" content="text/html>
   <meta charset="utf-8">
    <title>D3 Page Template</title>
    <script type="text/javascript" src="d3/d3.js" charset="utf-8"></script>
  </head>
  <body>
      <script type="text/javascript">
        // TODO
      </script>
  </body>
</html>
```

D3: Setup

- Running a Python mini web server:

Python 2.x:

```
python –m SimpleHTTPServer 8080
```

Python 3.x:

```
python –m http.server 8080
```

You should get:

```
127.0.0.1 - - [02/Dec/2024 22:58:35] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [02/Dec/2024 22:58:35] "GET /d3/d3.js HTTP/1.1" 200 -
```