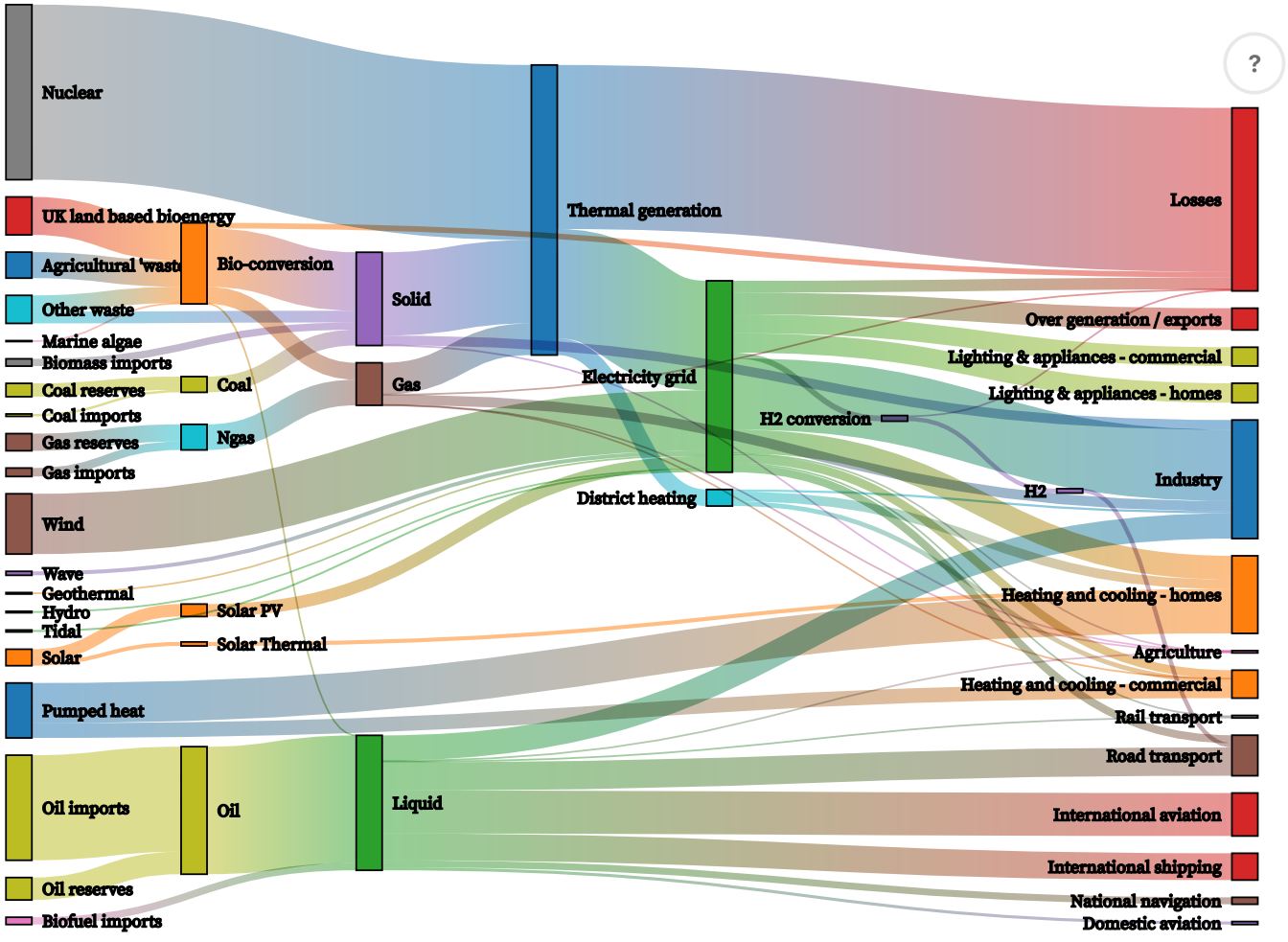
Xiang Gao • geekplux.com
Full-time Learner.

10

Published Jun 2, 2018
Fork of Sankey Diagram by D3 • 1 Fork

Draggable D3 Sankey Diagram




```
chart = {
  const svg = d3.select(DOM.svg(width, height));

  const graph = sankey(data);

  const node = svg.append("g")
      .attr("stroke", "#000")
      .selectAll("g")
      .data(graph.nodes)
      .enter()
      .append('g')
      .attr('transform', d => `translate(${d.x0}, ${d.y0})`)

  node.append("rect")
      .attr("height", d => d.y1 - d.y0)
```

```
      .attr("width", d => d.x1 - d.x0)
      .attr("fill", d => color(d.name));

  node.append("text")
      .attr("x", d => d.x0 < width / 2 ? 6 + (d.x1 - d.x0) : - 6)
      .attr("y", d => (d.y1 - d.y0) / 2)
      .attr("dy", "0.35em")
      .attr("text-anchor", d => d.x0 < width / 2 ? "start" : "end")
      .attr('font-size', 10)
      .text(d => d.name);

   node
     .attr('cursor', 'move')
     .call(d3.drag()
           .on('start', dragStart)
           .on('drag', dragMove)
           .on('end', dragEnd));

   const link = svg.append("g")
      .attr("fill", "none")
      .attr("stroke-opacity", 0.5)
     .selectAll("g")
     .data(graph.links)
     .enter().append("g")
       .style("mix-blend-mode", "multiply");

   const gradient = link.append("linearGradient")
      .attr("id", d => (d.uid = DOM.uid("link")).id)
      .attr("gradientUnits", "userSpaceOnUse")
      .attr("x1", d => d.source.x1)
      .attr("x2", d => d.target.x0);

   gradient.append("stop")
      .attr("offset", "0%")
      .attr("stop-color", d => color(d.source.name));

   gradient.append("stop")
      .attr("offset", "100%")
      .attr("stop-color", d => color(d.target.name));

   const path = link.append("path")
      .attr("d", d3.sankeyLinkHorizontal())
      .attr("stroke", d => d.uid)
      .attr("stroke-width", d => Math.max(1, d.width));

   function dragStart (d) {
     // this.parentNode.appendChild(this);
     // if (this.nextSibling) this.parentNode.appendChild(this);

     if (!d.__x) d.__x = d3.event.x;
     if (!d.__y) d.__y = d3.event.y;
     if (!d.__x0) d.__x0 = d.x0;
     if (!d.__y0) d.__y0 = d.y0;
     if (!d.__x1) d.__x1 = d.x1;
     if (!d.__y1) d.__y1 = d.y1;
```

```
  }

  function dragMove(d) {
    d3.select(this)
      .attr('transform', function (d) {
      const dx = d3.event.x - d.__x;
      const dy = d3.event.y - d.__y;
      d.x0 = d.__x0 + dx;
      d.x1 = d.__x1 + dx;
      d.y0 = d.__y0 + dy;
      d.y1 = d.__y1 + dy;

      // 防止超出边界
      if (d.x0 < 0) {
        d.x0 = 0;
        d.x1 = nodeWidth;
      }
      if (d.x1 > width) {
        d.x0 = width - nodeWidth;
        d.x1 = width;
      }
      if (d.y0 < 0) {
        d.y0 = 0;
        d.y1 = d.__y1 - d.__y0;
      }
      if (d.y1 > height) {
        d.y0 = height - (d.__y1 - d.__y0);
        d.y1 = height;
      }

      return `translate(${d.x0}, ${d.y0})`;
    });
      sankey.update(graph);
      path.attr('d', d3.sankeyLinkHorizontal());
  }

  function dragEnd(d) {
    delete d.__x;
    delete d.__y;
    delete d.__x0;
    delete d.__x1;
    delete d.__y0;
    delete d.__y1;
  }

  return svg.node();
}
```

Sankey diagrams visualize the magnitude of flow between nodes in a network. This intricate diagram shows a possible scenario for UK energy production and consumption in 2050: energy **supplies** are on the left, and **demands** are on the right. Intermediate nodes group related forms of production and show how energy is converted and transmitted before it is consumed or lost. The thickness of each link encodes the amount of flow from source to target.

This notebook is built with d3-sankey, which computes positions via iterative relaxation. After fixing the horizontal position of each node, the algorithm starts from the sources on the left, positioning downstream nodes so as to minimize link distance. A reverse pass is then made from right-to-left, and then the entire process is repeated several times. Overlapping nodes are shifted to avoid collision.

The fully automatic layout is convenient for rapid visualization—positioning nodes manually is tedious! However, the algorithm is not perfect; it could be improved to minimize link crossings and to prevent links from intersecting unrelated nodes. It also does not support cyclical networks.

Data: Department of Energy & Climate Change, Tom Counsell

```
nodeWidth = 15
```

```
sankey = ƒ()
```

```
format = ƒ(d)
```

```
color = ƒ(name)
```

```
data = ▶ Object {nodes: Array(48), links: Array(68)}
```

```
height = 540
```

```
d3 = ▶ Object {clientPoint: ƒ(t, n), create: ƒ(t), creator: ƒ(t), customEvent: ƒ(n, e, r, i), l
```