Notebook D3.js

Dataset

Questo file contiene il dataset pre-elaborato in Python. L'ho allegato usando Shift-Comando-U; è anche possibile allegare un file trascinandolo e rilasciandolo nel riquadro degli allegati.

```
data = ▸Array(405) [Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Ob
```

```
data = FileAttachment("df_nan.csv").csv()
```

```
▸Array(405) [Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, O
```

```
data
```

```
import {table} from "@observablehq/inputs"
```

```
import {table} from "@observablehq/inputs"
```

CLASSE DI LAUREA	NOME_CORSO	DIPARTIMENTO
L-1 - beni culturali	Scienze dei beni culturali	Ricerca e innovazione umanistica (diri
L-1 - beni culturali	Scienze dei beni culturali	Ricerca e innovazione umanistica (diri
L-1 - beni culturali	Scienze dei beni culturali	Ricerca e innovazione umanistica (diri
L-10 - lettere	Lettere	Ricerca e innovazione umanistica (diri
L-10 - lettere	Lettere	Ricerca e innovazione umanistica (diri
L-10 - lettere	Lettere	Ricerca e innovazione umanistica (diri
L-11 - lingue e culture moderne	Lingue, culture e letterature moderne	Ricerca e innovazione umanistica (diri
L-11 - lingue e culture moderne	Lingue, culture e letterature moderne	Ricerca e innovazione umanistica (diri
L-11 - lingue e culture moderne	Lingue, culture e letterature moderne	Ricerca e innovazione umanistica (diri
L-12 - mediazione linguistica	Lingue e culture per il turismo e la mediazione internazionale	Ricerca e innovazione umanistica (diri
L-12 - mediazione linguistica	Lingue e culture per il turismo e la mediazione internazionale	Ricerca e innovazione umanistica (diri

```
viewof myTable = Inputs.table(data)
```

Introduzione di `Observable`

Scales

Le scale D3 sono disponibili in molti tipi. La scelta dipende dalla dimensione astratta (quantitativa o nominale) e dalla variabile visiva (posizione o colore).

SVG e Canvas sono generici, consentono qualsiasi tipo di grafica. D3, invece, è pensato per la visualizzazione e quindi fornisce un vocabolario specializzato di forme, ovvero funzioni che generano dati di percorso.

Animazioni

Un'animazione non è un singolo grafico, ma una sequenza di grafici nel tempo. Questa sequenza può essere rappresentata come una cella (o funzione) che restituisce il grafico per un dato tempo t. Per semplicità, spesso utilizziamo il tempo normalizzato, dove $t = 0$ è l'inizio dell'animazione e $t = 1$ è la fine.

Per facilitare l'animazione (tra gli altri usi), D3 fornisce degli interpolatori. Il più generico di questi, `d3.interpolate`, accetta numeri, colori, stringhe di numeri e persino array e oggetti. Dato un valore iniziale e uno finale, `d3.interpolate` restituisce una funzione che accetta un tempo $0 \le t \le 1$ e restituisce il valore intermedio corrispondente.

Tuttavia, l'animazione è più di una semplice interpolazione: è anche una questione di tempistica. Dobbiamo ridisegnare sessanta volte al secondo e calcolare il tempo normalizzato t in base al tempo reale, all'ora di inizio desiderata e alla durata dell'animazione.

Testo

Il testo è una selezione di elementi di testo, inizialmente vuoti, il cui elemento "padre" è l'elemento SVG. Questo elemento padre determina dove verranno aggiunti in seguito gli elementi di testo in ingresso.

Chiamando `selection.data`, il testo viene associato a un nuovo array di dati, letters. Questo calcola tre sottoinsiemi disgiunti della selezione di testo: la selezione di ingresso che rappresenta i nuovi dati per i quali non esiste alcun elemento; la selezione di aggiornamento che rappresenta gli elementi esistenti per i quali sono presenti nuovi dati; e la selezione di uscita che rappresenta gli elementi esistenti per i quali non sono presenti nuovi dati.

Ci sono quasi sempre troppe informazioni rispetto a quelle che possono ragionevolmente "entrare" in un'immagine. Quindi il design non consiste solo nel decidere come mostrare qualcosa, ma cosa mostrare e cosa non mostrare in base a ciò che riteniamo importante per il lettore immaginario.

E grazie ai computer, il lettore reale può ora avere voce in capitolo: l'immagine può essere personalizzata su richiesta in base ai suoi interessi.

Eppure questo potere è un'arma a doppio taglio. L'interattività consente al lettore di far emergere più informazioni, ma lo costringe a impegnarsi per ottenerle. Se non stiamo attenti, potremmo nascondere informazioni importanti dietro controlli su cui i lettori non cliccano mai.

Mantra di Shneiderman

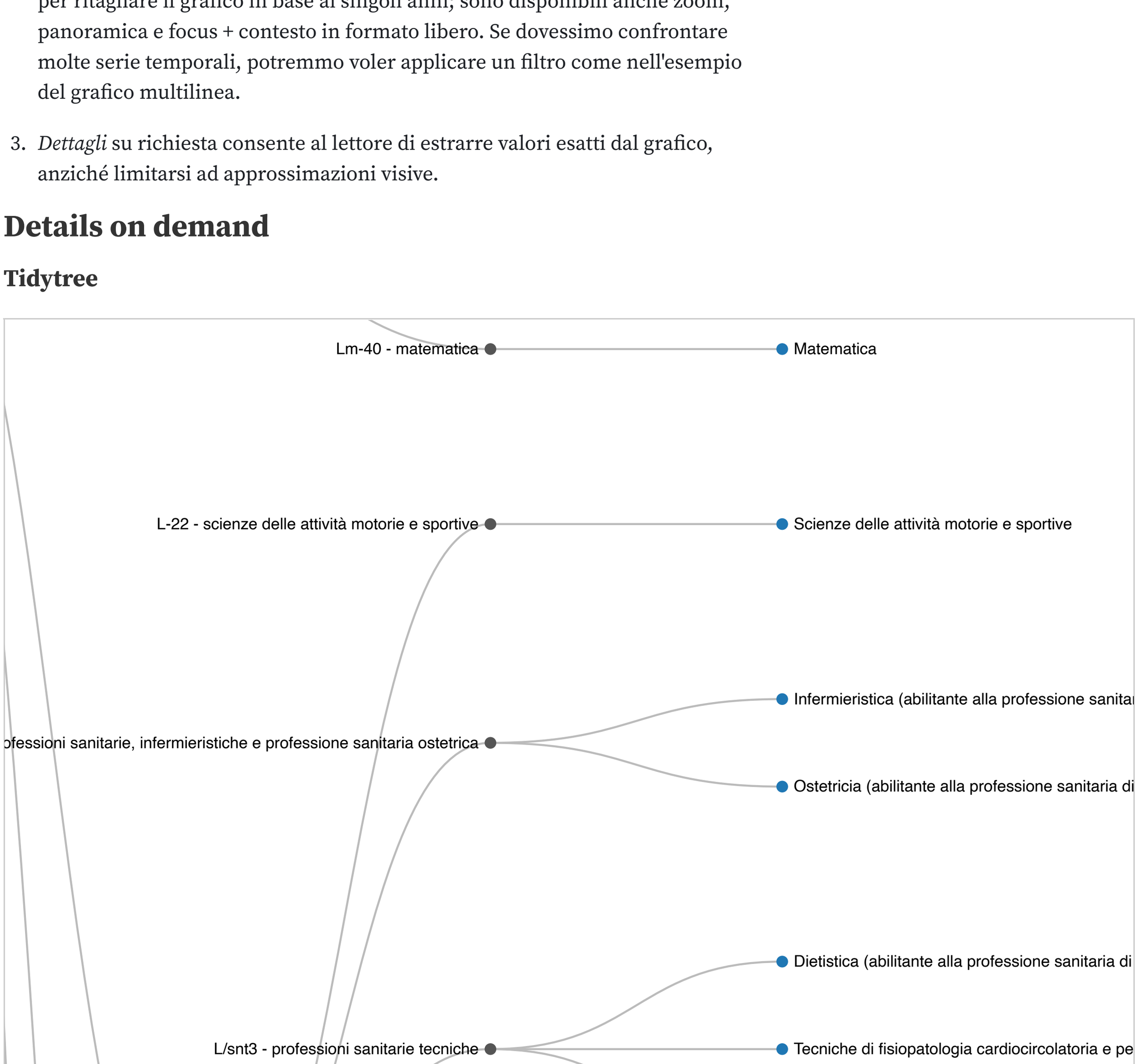
Una buona linea guida per l'interazione è il mantra di Ben *Shneiderman* per la ricerca di informazioni:

Overview first,
zoom and filter,
then details on demand.

1. La *panoramica* (Overview) è la forma iniziale del grafico. Il suo scopo non è mostrare tutto (cosa impossibile), ma fornire una visione "macro" di tutti i dati. La panoramica è una mappa che guida l'esplorazione del lettore.
2. Lo *zoom* e il filtro sono metodi per selezionare ciò che viene mostrato e concentrarsi su un argomento di interesse. In D3 sono disponibili i controlli per ritagliare il grafico in base ai singoli anni; sono disponibili anche zoom, panoramica e focus + contesto in formato libero. Se dovessimo confrontare molte serie temporali, potremmo voler applicare un filtro come nell'esempio del grafico multilinea.
3. *Dettagli* su richiesta consente al lettore di estrarre valori esatti dal grafico, anziché limitarsi ad approssimazioni visive.

Details on demand

Tidytree




```
(function() {

  /**
   * PREPARAZIONE E FILTRAGGIO
   * Selezioniamo solo i dati relativi ai Corsi di Studio (CDS) per evitare
   * di inserire indicatori aggregati che sporcherebbero la gerarchia.
   */
  const cdsData = data.filter(d => d.INDICATORE_TYPE === "CDS");

  /**
   * COSTRUZIONE DELLA GERARCHIA (DATA MAPPING)
   * Questa funzione trasforma i dati piatti in una struttura a "nidi".
   */
  function buildHierarchy(data) {
    const root = { name: "Università", children: [] };
    const depMap = new Map();

    data.forEach(d => {
      const dep = d.DIPARTIMENTO;
      const classe = d["CLASSE DI LAUREA"];
      const corso = d.NOME_CORSO;
      const value = +d.Percentuale_Occupazione_1anno || 0;

      if (!depMap.has(dep)) {
        depMap.set(dep, { name: dep, children: [], classMap: new Map() });
        root.children.push(depMap.get(dep));
      }

      const depObj = depMap.get(dep);

      if (!depObj.classMap.has(classe)) {
        depObj.classMap.set(classe, { name: classe, children: [] });
        depObj.children.push(depObj.classMap.get(classe));
      }

      // Aggiungiamo il singolo corso come nodo finale (foglia)
      depObj.classMap.get(classe).children.push({ name: corso, value });
    });

    // Pulizia dei riferimenti temporanei
    root.children.forEach(dep => dep.classMap = null);
    return root;
  }

  const hierarchyData = buildHierarchy(cdsData);

  /**
   * CALCOLO DEL LAYOUT DELL'ALBERO
   * d3.hierarchy analizza la struttura dati.
   * d3.tree().nodeSize([altezza, larghezza]) definisce quanto spazio
   * deve esserci tra un nodo e l'altro (fondamentale per evitare sovrapposizioni).
   */
  const root = d3.hierarchy(hierarchyData)
    .sort((a, b) => d3.ascending(a.data.name, b.data.name)); // Ordina alfabeticamente

  const treeLayout = d3.tree().nodeSize([60, 200]);
  treeLayout(root);

  /**
   * CREAZIONE DEL CANVAS SVG
   * Usiamo dimensioni molto grandi (10000x10000) perché un albero universitario
   * può essere enorme. L'utente lo esplorerà tramite lo zoom/trascinamento.
   */
  const svg = d3.create("svg")
    .attr("width", 10000)
    .attr("height", 10000)
    .style("font", "11px sans-serif")
    .style("cursor", "grab");

  // Gruppo contenitore centrato per permettere il panning
  const container = svg.append("g")
    .attr("transform", "translate(5000, 500)");

  /**
   * DISEGNO DEI COLLEGAMENTI (LINKS)
   * Crea i percorsi curvi (archi) che collegano i genitori ai figli.
   * d3.linkHorizontal() è perfetto per alberi che si sviluppano da sinistra a destra.
   */
  container.selectAll("path.link")
    .data(root.links())
    .join("path")
      .attr("fill", "none")
      .attr("stroke", "#bbb")
      .attr("stroke-width", 1.4)
      .attr("d", d3.linkHorizontal()
        .x(d => d.y) // Scambiamo X e Y per lo sviluppo orizzontale
        .y(d => d.x)
      );

  /**
   * DISEGNO DEI NODI E DEI TESTI
   * Posizioniamo un pallino per ogni elemento della gerarchia.
   */
  const node = container.selectAll("g.node")
    .data(root.descendants())
    .join("g")
      .attr("class", "node")
      .attr("transform", d => `translate(${d.y},${d.x})`);

  // I nodi con figli sono grigi, i corsi finali sono blu
  node.append("circle")
    .attr("r", 4)
    .attr("fill", d => d.children ? "#555" : "#1f77b4");

  // Allineamento intelligente del testo: a sinistra se ha figli, a destra se è una foglia
  node.append("text")
    .attr("dy", "0.32em")
    .attr("x", d => d.children ? -8 : 8)
    .attr("text-anchor", d => d.children ? "end" : "start")
    .style("font-weight", d => d.depth === 1 ? "600" : "400")
    .text(d => d.data.name);

  /**
   * INTERATTIVITÀ (PAN + ZOOM)
   * Permette all'utente di navigare nell'albero usando la rotella del mouse
   * o trascinando con il click (stile Google Maps).
   */
  svg.call(
    d3.zoom()
      .scaleExtent([0.1, 3]) // Limiti di zoom (da 10% a 300%)
      .on("zoom", (event) => container.attr("transform", event.transform))
  );

  /**
   * RENDERING FINALE
   * Inseriamo l'SVG dentro un contenitore HTML con overflow nascosto per creare
   * una "finestra" di visualizzazione pulita sulla pagina.
   */
  return html`<div style="
      width: 100%;
      height: 700px;
      overflow: hidden;
      border: 1px solid #ccc;
    ">
    ${svg.node()}
  </div>`;

})();
```

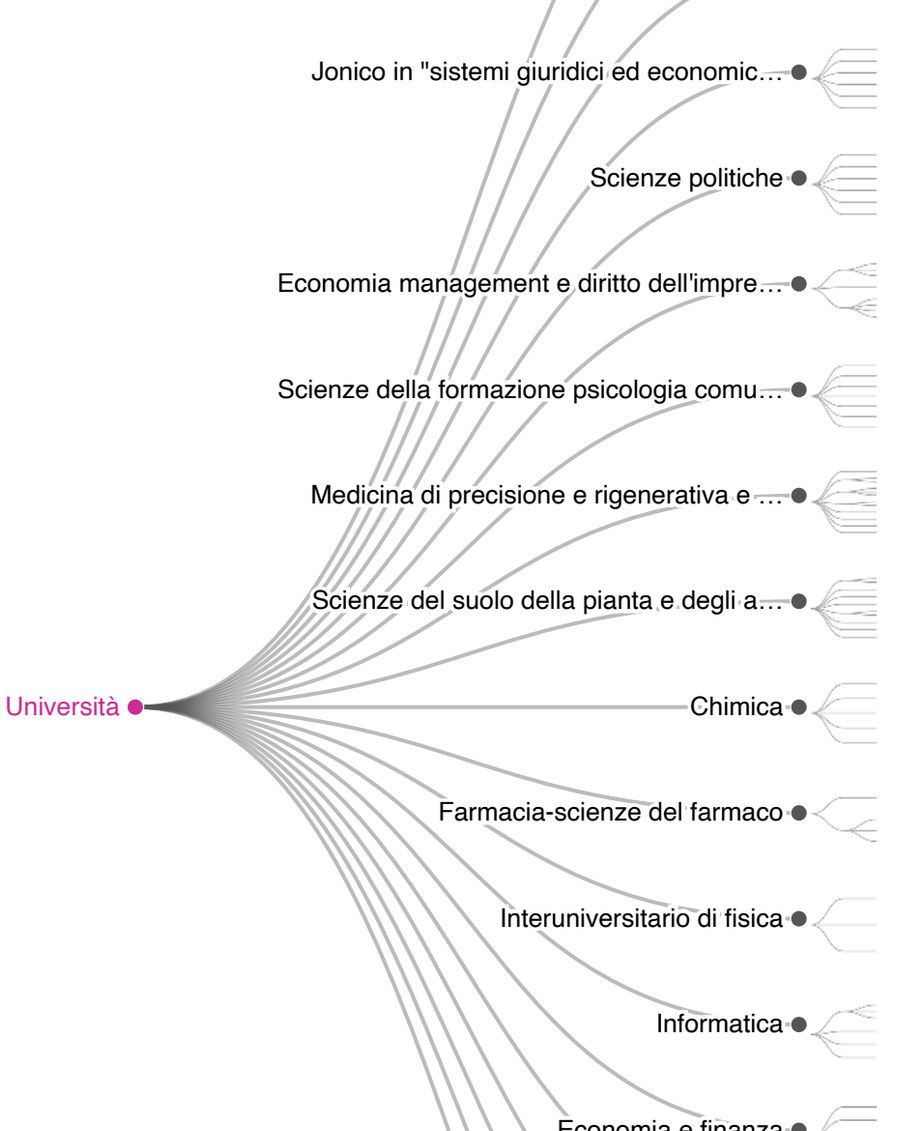
Space tree

```
import {SpaceTree} from "@john-guerra/spacetree"
```

```
import {SpaceTree} from "@john-guerra/spacetree"
```

```
buildHierarchy = ƒ(data)
```

```
function buildHierarchy(data) {
  const root = { name: "Università", value: 1, children: [] };
  const depMap = new Map();

  const shorten = str =>
    (str && str.length > 40 ? str.slice(0, 40) + "…" : str);

  data.forEach(d => {
    const dep = d.DIPARTIMENTO;
    const classe = d["CLASSE DI LAUREA"];
    const corso = d.NOME_CORSO;

    // Department
    if (!depMap.has(dep)) {
      depMap.set(dep, {
        name: dep,
        labelShort: shorten(dep),
        value: 1,
        children: [],
        classMap: new Map()
      });
      root.children.push(depMap.get(dep));
    }

    const depObj = depMap.get(dep);

    // Class
    if (!depObj.classMap.has(classe)) {
      depObj.classMap.set(classe, {
        name: classe,
        labelShort: shorten(classe),
        value: 1,
        children: []
      });
      depObj.children.push(depObj.classMap.get(classe));
    }

    // Course (leaf)
    depObj.classMap.get(classe).children.push({
      name: corso,
      labelShort: shorten(corso),
      labelFull: corso,
      value: 1,              // ← REQUIRED
      Categoria_laurea: d.Categoria_laurea
    });
  });

  root.children.forEach(dep => dep.classMap = null);
  return root;
}
```




```
viewof knowledgeChiSelected = SpaceTree(
  buildHierarchy(data.filter(d => d.INDICATORE_TYPE === "CDS")),
  {
    label: d => d.data.labelShort || d.data.name,
    nodeTitle: d => d.data.labelFull || d.data.name,
    width: 1200,
    height: 800,
    collapsible: true,
    zoom: true,
    linkColor: "#888"
  }
)
```

Riferimenti:

1. *Link per ulteriori argomenti dai tutorial di Observable*: https://observablehq.com/@d3/learn-d3-further-topics?collection=@d3/learn-d3
2. *Link ufficiale per d3*: https://d3js.org/what-is-d3
3. *Link per tutorials e galleria di esempi:* https://observablehq.com/@d3/gallery?utm_source=d3js-org&utm_medium=page-nav&utm_campaign=try-observable
4. *Link di Spacetree*: https://observablehq.com/@john-guerra/spacetree