Cruzeiro do Sul Virtual
Educação a Distância

VISUALIZAÇÃO HIERÁRQUICA

Prof. Ismar Frango

Visualização De Informação Hierárquica

Em muitas situações na área de Ciência de Dados, as informações a serem processadas e visualizadas podem ter natureza **hierárquica**.

Dados hierárquicos são geralmente exibidos em forma de gráficos de **árvore**, chamados assim devido à sua semelhança com a estrutura de uma árvore de cabeça para baixo, com a raiz no topo e os galhos abaixo dela.

Uma hierarquia sempre começa com uma **raiz**, que pode representar uma pasta em um sistema de arquivos, o presidente de uma organização, etc. A raiz possui pelo menos um elemento hierarquicamente dependente dele, chamado de nó-filho – nesse caso, a raiz é o nó-pai. Cada nó-filho pode ter zero ou mais filhos. Desta maneira, um *dataset* representa um conjunto de informações hierárquicas se ele é organizado da seguinte forma: todos os elementos do dataset são subordinados **diretamente** a apenas um elemento (têm apenas um nó-pai); um nó-pai pode ter vários filhos. A única exceção é o nó raiz, que não tem nenhum nó-pai. O elemento que não possui galhos é conhecido como folha ou nó terminal.

Um exemplo de informações hierárquicas pode ser visto a seguir:

`employee` table

EmpNo	First Name	Last Name	Dept. Num
100	Mahwish	Faki	10-L
101	Hamadh	Hashim	10-L
102	Nirun	Ar	20-B
103	Chaaya	Sandakelum	20-B

`computer` table

Serial Num	Type	User EmpNo
3009734-4	Computer	100
3-23-283742	Monitor	100
2-22-723423	Monitor	100
232342	Printer	100

Fonte: https://en.wikipedia.org/wiki/Hierarchical_database_model - Licença: CC-BY

Fonte: https://storage.needpix.com/rsynced_images/roots-153966_1280.png - Licença: CC-BY

Os dados acima podem ser representados pela seguinte árvore (a **raiz** está representada por */*):

Fonte: Autor + GraphOnline

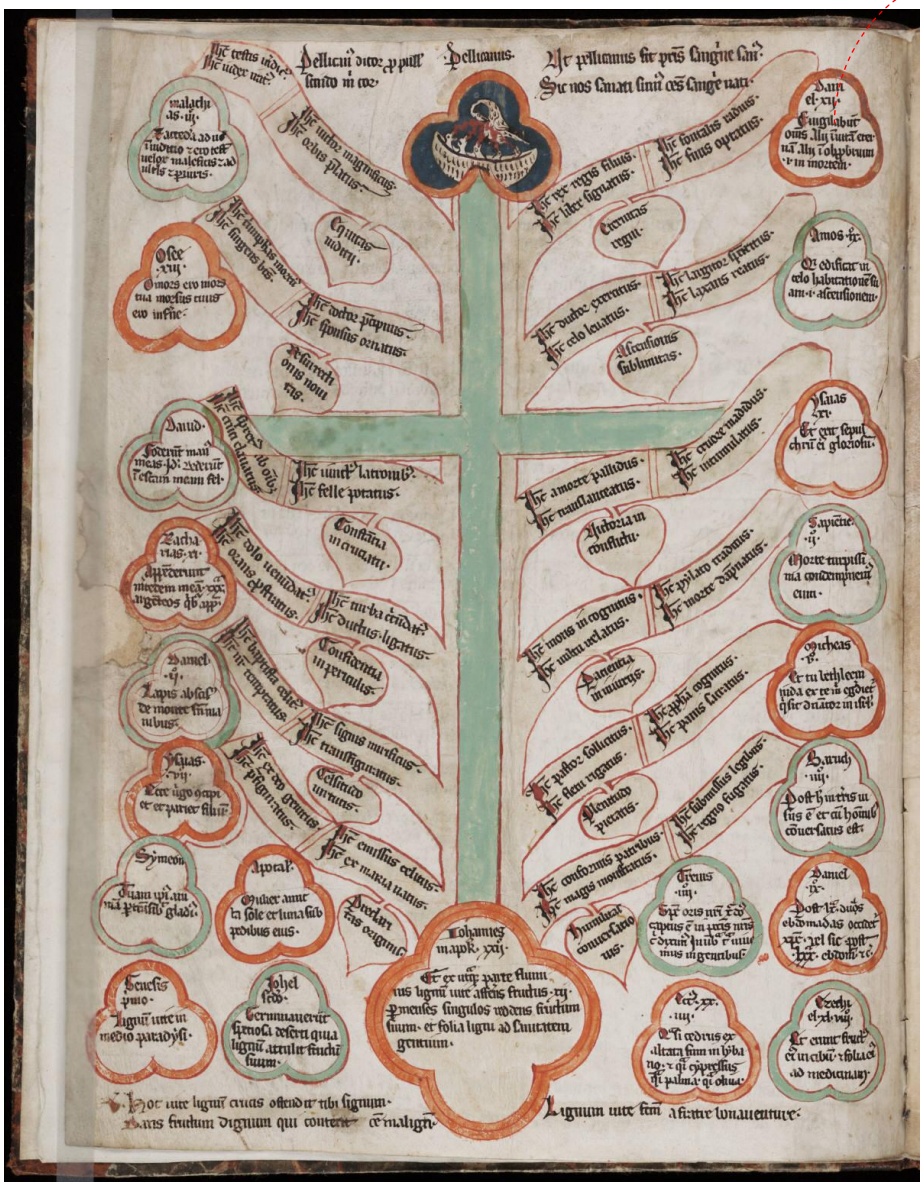
Cruzeiro do Sul Virtual
Educação a Distância

Um pouco de História

Conheça um pouco mais sobre essa figura em: http://brbl-archive.library.yale.edu/exhibitions/speculum/1v-tree-of-life.html

Livros antigos mostram o uso de representações de árvores para indicar relações hierárquicas (geralmente relações de poder ou familiares)

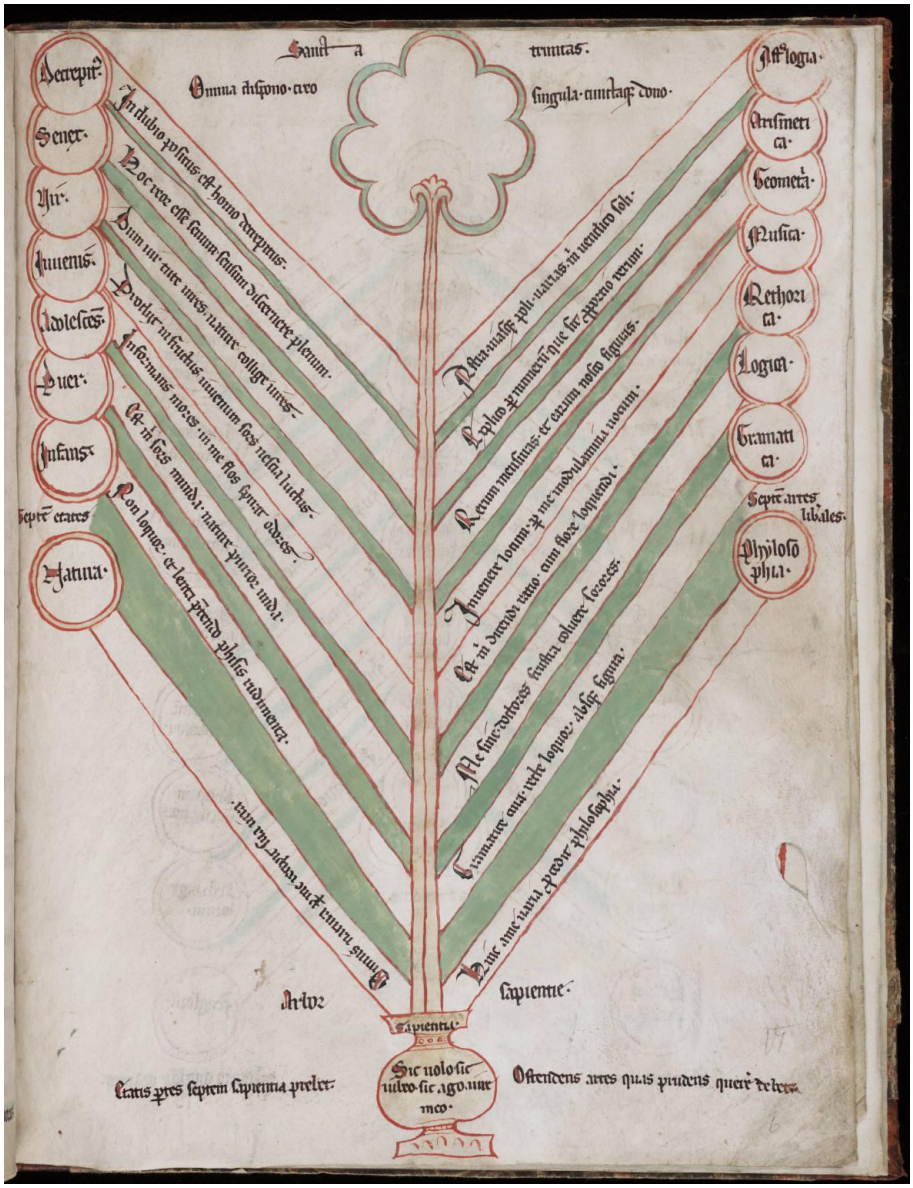
Um exemplo é a “Árvore da Vida”, uma figura do século XIII:



Fonte: http://brbl-archive.library.yale.edu/exhibitions/speculum/

Este diagrama apresenta os eventos da vida de Jesus Cristo, juntamente com citações das escrituras e instruções para meditação na forma de uma árvore. A árvore tem doze ramos e doze frutos (nós), cada um apresentando um mistério diferente da vida de Cristo.

Uma outra árvore que aparece em uma obra antiga (séc XIII) é a “Árvore da Sabedoria”: também com forte influência cristã, este diagrama mostra as sete “idades” do homem (primeira infância, infância, adolescência, juventude, adulto, madureza e velhice) à esquerda e as sete artes liberais da época à direita (gramática, lógica, retórica, música, geometria, aritmética e astrologia), culminando na Santíssima Trindade no topo. As sete idades são colocadas na categoria "Natureza". As artes liberais têm "Filosofia" como categoria principal.



Fonte: http://brbl-archive.library.yale.edu/exhibitions/speculum/

Algumas aplicações

Dados organizados de maneira hierárquica aparecem em muitas situações práticas do dia-a-dia. Por exemplo, uma aplicação bastante frequente nas áreas de Administração e Recursos Humanos é a elaboração de **mapas mentais**. Um mapa mental é um diagrama de árvore voltado para a gestão de informações, de conhecimento. Sistematizado por Tony Buzan, é amplamente utilizado para registrar *brainstormings,* na compreensão e descrição de problemas e como ferramenta de aprendizagem, por exemplo.

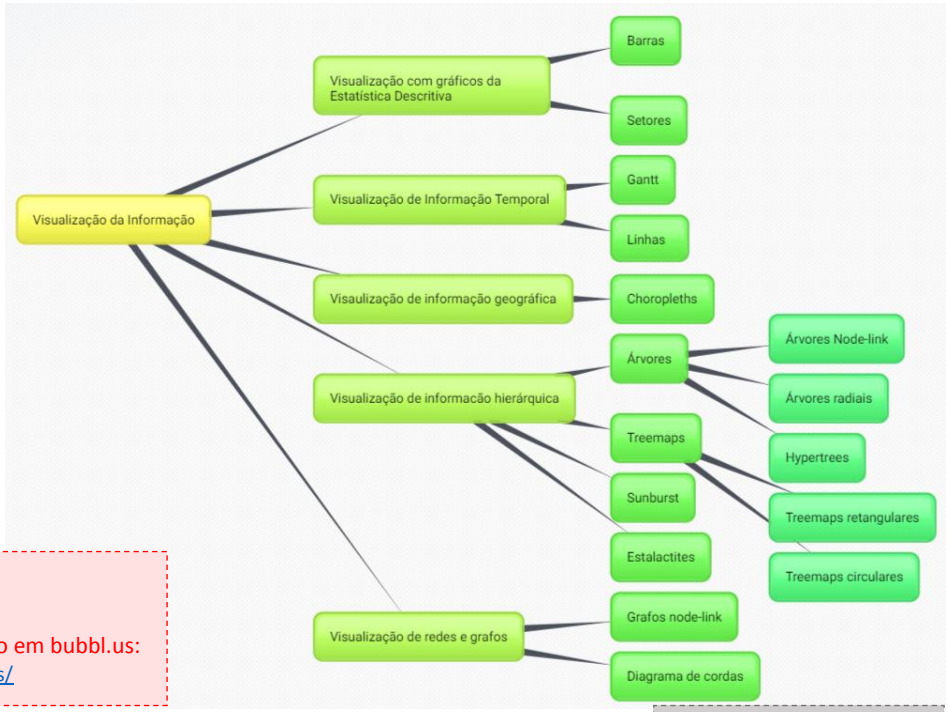
Conheça um pouco mais sobre ele em: https://pt.wikipedia.org/wiki/Tony_Buzan

Fonte: https://pt.m.wikipedia.org/wiki/Ficheiro:Memorize_mind_map.png – Licença CC-BY

Existem várias ferramentas de geração de mapas mentais disponíveis na Internet, como:

- bubbl.us,
- Mindmeister,
- Text2MindMap,

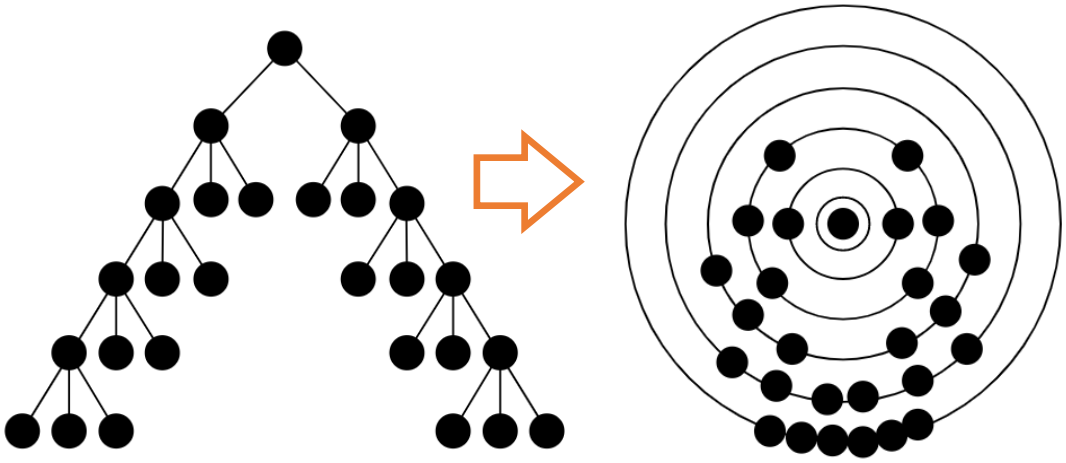
entre outras.



Esse mapa mental foi feito em bubbl.us: http://bubll.us/

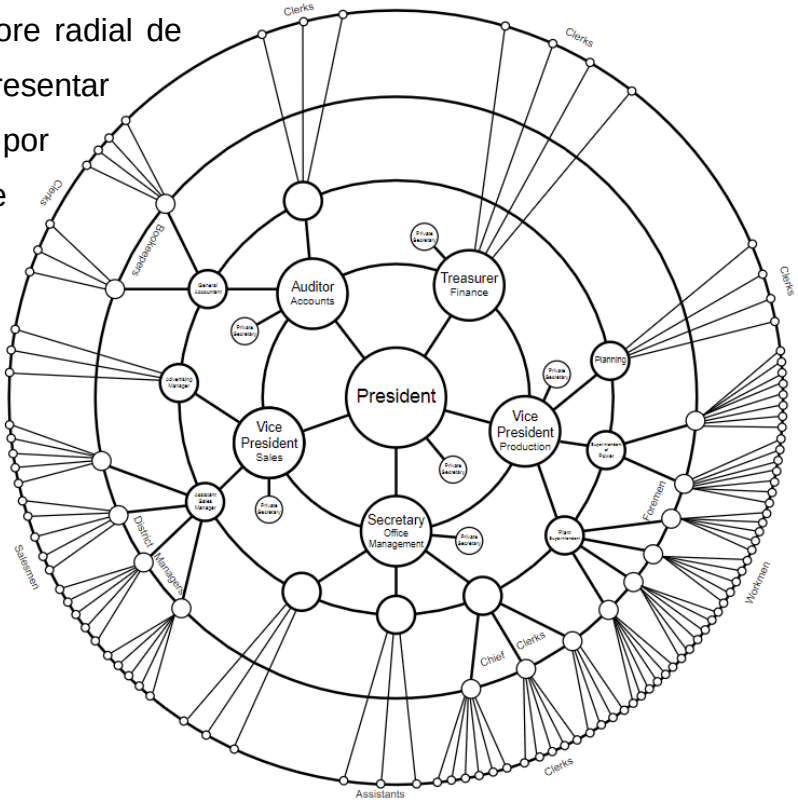
Fonte: Autor (usando bubbl.us)

Um mapa mental nada mais é que uma árvore do tipo *node-link tree*, ou seja, que exibe apenas os nós (elementos) e suas ligações. Esse tipo de árvore pode ser disposto como nos exemplos acima (chamada de organização triangular), porém corre o risco de ficar muito larga (caso em que cada nó tem muitos filhos) ou muito alta (caso em que há muitas hierarquias de nós). Uma solução para melhorar a visualização é usar uma organização radial:



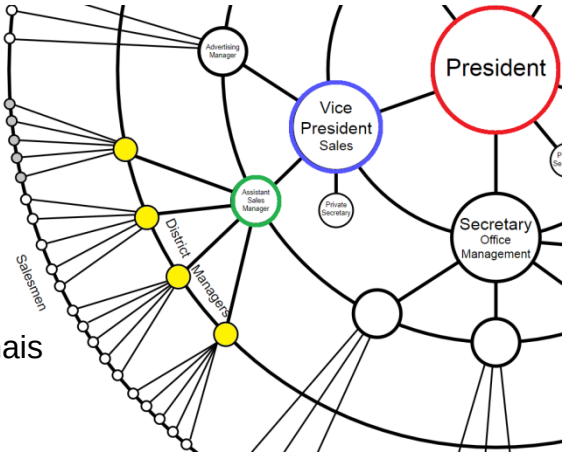
Fonte: https://en.wikipedia.org/wiki/Radial_tree#/media/File:Radial-vs-tri.svg – Licença CC-BY. Alterada pelo autor

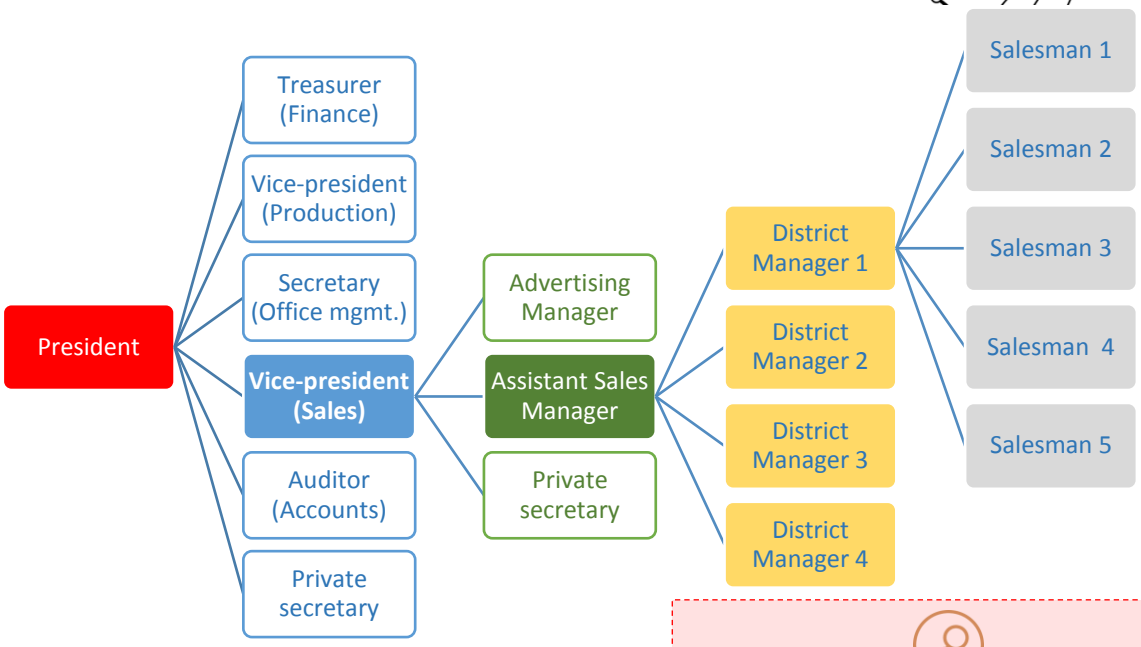
Segue o exemplo de uma árvore radial de um organograma, usado para representar cargos em uma empresa, por exemplo. Note que é uma árvore que não é tão profunda (contando a partir da raiz, tem-se quatro níveis apenas). Entretanto, é uma árvore muito larga, especialmente devido ao último nível, que tem muitos nós ("folhas").



Fonte: https://en.wikipedia.org/wiki/Radial_tree#/media/File:Radial_tree_-_Graphic_Statistics_in_Management.svg – Licença CC-BY

O exemplo a seguir mostra uma vizualização alternativa de um pequeno trecho da árvore radial (conforme seleção indicada ao lado), como uma *node-link tree*. Veja que somente uma parte foi representada no diagrama *node-link* convencional (triangular) pois esse tipo de visualização tende a ocupar bastante mais espaço do que a radial.

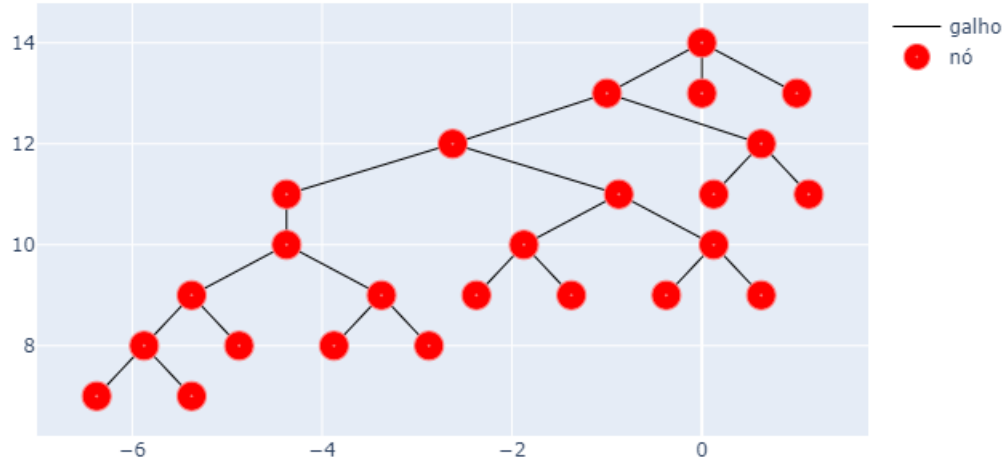




Um interessante mecanismo alternativo de visualização de árvores são as *Hyperbolic Trees*, ou *Hypertrees*. Um exemplo de aplicação: https://hyperbolic-tree-of-life.github.io/

Implementação em Python

O código a seguir apresenta uma implementação em Python que gera a seguinte árvore:



Note o uso das bibliotecas **plotly** (para exibição da árvore) e **igraph** (para representação interna) da árvore.

```
import plotly.graph_objects as go
import igraph
from igraph import Graph, EdgeSeq
nr_vertices = 25
v_label = list(map(str, range(nr_vertices)))
G = Graph.Tree(nr_vertices, 2) # 2 é o número de filhos por nó
lay = G.layout('rt')

position = {k: lay[k] for k in range(nr_vertices)}
Y = [lay[k][1] for k in range(nr_vertices)]
M = max(Y)
```

Lógica para organização dos nós na árvore

```
es = EdgeSeq(G) # sequence of edges
E = [e.tuple for e in G.es] # list of edges
L = len(position)
Xn = [position[k][0] for k in range(L)]
Yn = [2*M-position[k][1] for k in range(L)]
Xe = []
Ye = []
for edge in E:
    Xe+=[position[edge[0]][0],position[edge[1]][0], None]
    Ye+=[2*M-position[edge[0]][1],2*M-position[edge[1]][1], None]
```

O código usa dois gráficos do tipo *scatter* (um para exibir as linhas e um outro para exibir os nós)

```
labels = v_label
fig = go.Figure()
fig.add_trace(go.Scatter(x=Xe,
                         y=Ye,
                         mode='lines',
                         name='galho',
                         line=dict(color='rgb(0,0,0)', width=1),
                         hoverinfo='none'
                  ))
fig.add_trace(go.Scatter(x=Xn,
                  y=Yn,
                  mode='markers',
                  name='nó',
                  marker=dict(  symbol='circle-dot',
                                size=18,
                                color='#FF0000',
                                line=dict(color='rgb(255,127,127)',
                                width=1)
                                ),
                  text=labels,
                  hoverinfo='text',
                  opacity=1
                  ))
```

Note que não há uma opção de visualização de árvores do tipo *node link* na biblioteca plotly, sendo necessário utilizar gráficos do tipo *scatter*.

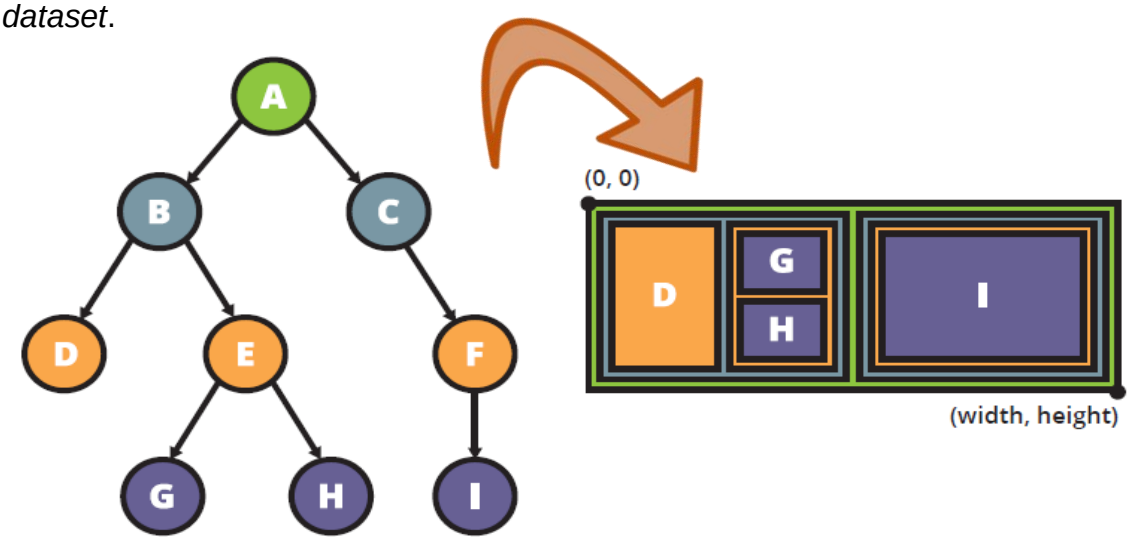
Uma outra opção é usando GraphViz com Python:
https://pypi.org/project/graphviz/

Representações visuais alternativas

Há outras possibilidades de exibição de informação hierárquica além das *node-link trees*. Uma delas é a visualização por Treemaps.

Um *treemap* é uma estratégia de visualização de informação hierárquica como um conjunto de retângulos aninhados. Para cada nó da árvore é definido um retângulo, que é, então, preenchido com retângulos menores, os quais representam nós-filhos. Cada retângulo pode ter uma área proporcional a um valor existente no *dataset*.

Treemaps foram propostos pela primeira vez em 1998 por Ben Shneiderman, importante cientista da área de Visualização da Informação. O artigo original (em inglês) encontra-se em: http://www.cs.umd.edu/hcil/treemap-history/index.shtml



Note que a possibilidade de se ter a representação de mais uma dimensão de informação numérica (por meio das áreas dos retângulos) faz com que a representação por *treemaps* tenha vantagens em relação à representação por *node-link trees*, além de ocupar menos espaço.

Exemplo: o dataset a seguir contém a população das 5 regiões do Brasil (em milhares):

Brasil		202.640
Centro-Oeste	Brasil	14.058
Nordeste	Brasil	53.082
Norte	Brasil	18.400
Sudeste	Brasil	87.500
Sul	Brasil	29.600

Fonte: https://sidra.ibge.gov.br/

Este *dataset* pode ser visualizado de diferentes maneiras:

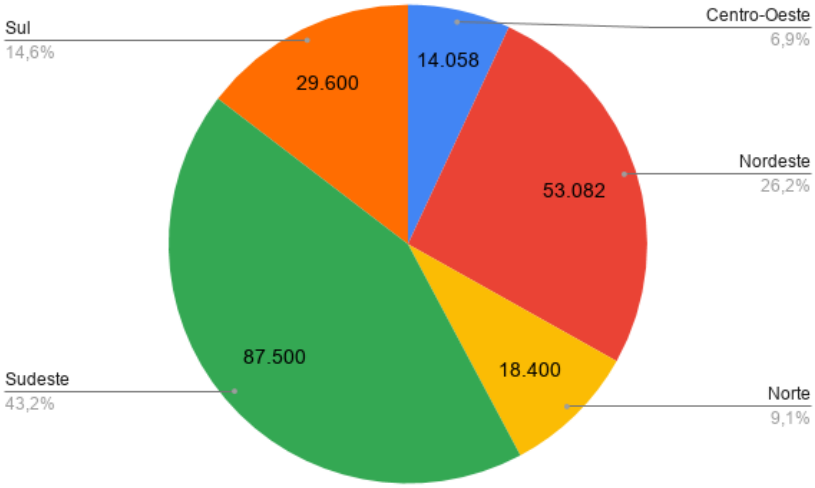
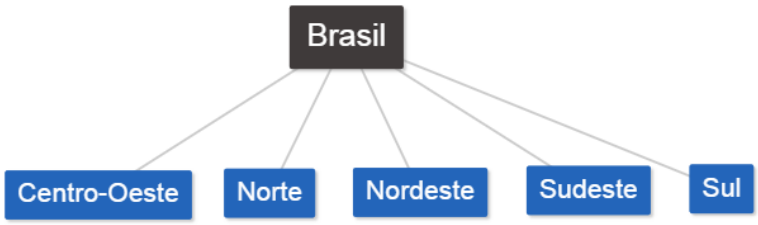


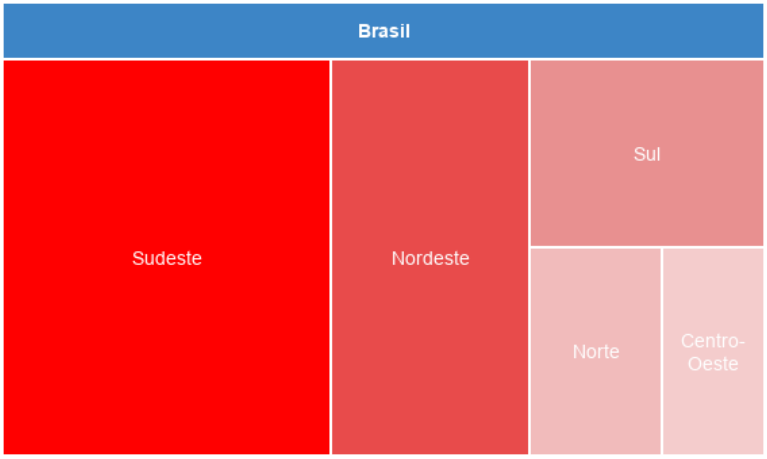
Gráfico de setores

É uma visualização bastante popular, embora tenha várias fragilidades: é difícil, visualmente, comparar as dimensões do setores sem rótulos de dados; além disso, quando o número de setores é grande, fica difícil identificar os setores por cores.
Gráfico feito com o Google Drive



Node-link tree

É fácil visualizar a organização hierárquica – porém, para hierarquias muito largas (muitos filhos por nó) ou profunda (muitos níveis), a visualização fica prejudicada. Além disso, as informações quantitativas só podem ser representadas textualmente.

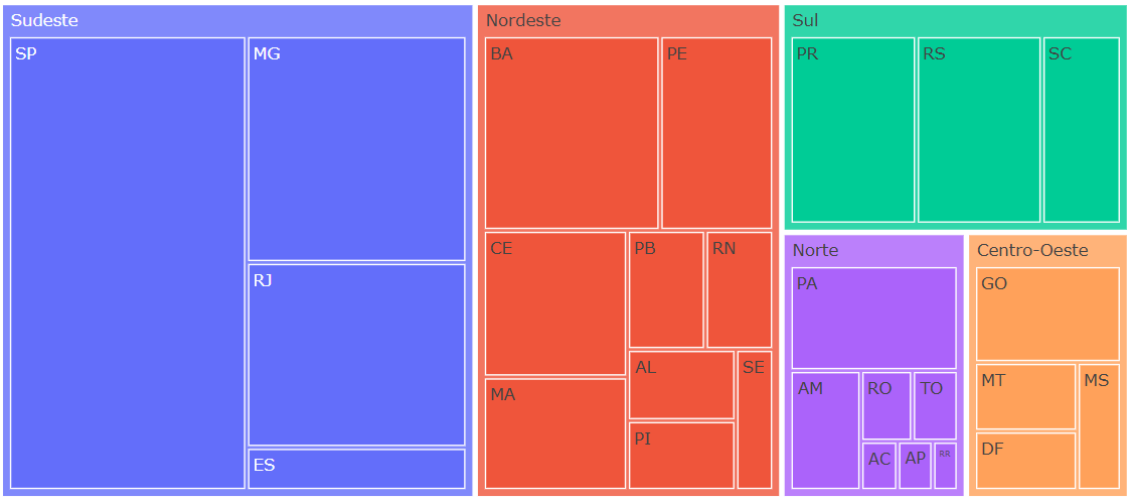


Treemap

É mais fácil de diferenciar tamanhos de áreas retangulares do que áreas de setores.de um círculo, o que faz a representação por Treemaps mais adequada que o Gráfico de Setores. Por representar hierarquias em um espaço mais reduzido e também informações quantitativas, é mais adequado que a representação *node-link*.
Gráfico feito com o Google Drive.

Nos *treemaps*, pode-se ainda utilizar outra dimensão de visualização, que é a cor, para representar alguma outra informação do *dataset*. Com isso, é possível, muitas vezes, encontrar padrões que seriam difíceis de perceber em outras estratégias de visualização.

O *treemap* do exemplo a seguir representa informações quantitativas sobre a população por estado no Brasil, organizados por regiões. Os tamanhos dos retângulos representam proporcionalmente as populações (tanto dos estados como das regiões) e as cores permitem identificar rapidamente cada região.



Note que os outros tipos de representação visual do exemplo anterior não seriam suficientes para representar as informações de hierarquia – o gráfico de setores não representa hierarquia – ou quantitativas – uma representação visual do tipo *node-link tree* poderia ser usada para mostrar três níveis de informação: Brasil → Regiões → Estados, mas a informação sobre a população dos estados teria que ser representada como texto.

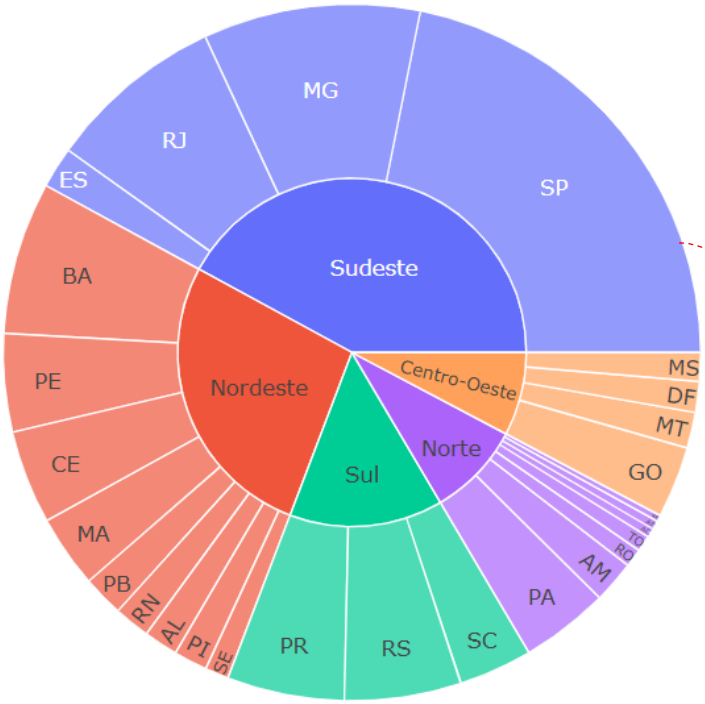
Esta visualização foi gerada com o código Python abaixo:

Organização do *dataset*. Claramente, ele pode vir de um arquivo .csv, por exemplo

```
import plotly.express as px
import pandas as pd

estados = ["RO","AC","AM","RR","PA","AP","TO","MA","PI","CE","RN","PB","PE",
"AL","SE","BA","MG","ES","RJ","SP","PR","SC","RS","MS","MT","GO","DF"]
regioes =
["Norte","Norte","Norte","Norte","Norte","Norte","Norte","Nordeste","Nordeste"
,"Nordeste","Nordeste","Nordeste","Nordeste","Nordeste","Nordeste","Nordeste",
"Sudeste","Sudeste","Sudeste","Sudeste","Sul","Sul","Sul","Centro-Oeste",
"Centro-Oeste","Centro-Oeste","Centro-Oeste"]
populacao = [1777225,881935,4144597,605761,8602865,845731,1572866,7075181,
3273227,9132078,3506853,4018127,9557071,3337357,2298696,14873064,21168791,4018650,172649
43,45919049,11433957,7164788,11377239,2778986,3484466,7018354,3015268]

df = pd.DataFrame(dict(estados=estados,regioes=regioes,populacao=populacao))
df["all"] = "all" #garante um único nó raiz

fig = px.treemap(df,path=[regioes,estados],values=populacao)
fig.show()
```

Saiba mais como gerar *treemaps* com o pacote plotly.express: https://plotly.com/python/treemaps/

Uma visualização alternativa aos treemaps, que guarda semelhanças com os gráficos de setores, mas com suporte à representação também da organização hierárquica, é o gráfico conhecido como *Sunburst*.



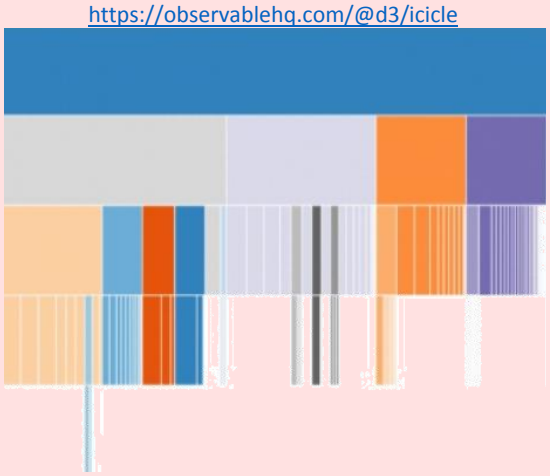
Também conhecido por gráfico polar, foi criado pela estatística e enfermeira inglesa Florence Nightingale em 1860: https://en.wikipedia.org/wiki/Pie_chart#/media/File:Nightingale-mortality.jpg

Este gráfico foi gerado com o mesmo código anterior, substituindo a chamada do método **treemap** por: px.**sunburst**

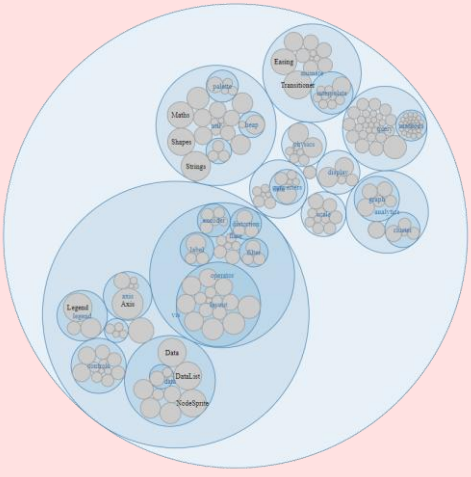
Conheça mais sobre como gerar *sunbursts* com Python em: https://plotly.com/python/sunburst-charts/

Entre outras possibilidades de visualização de informação hierárquica estão gráficos como os *treemaps* circulares (também conhecidos como *circle packs*) e os *icicles* (ou estalactites).

Conheça mais sobre os *icicles*: https://observablehq.com/@d3/icicle



Conheça mais sobre os *circle packs*: https://nextjournal.com/eda/interactive-circle-packing-plots



Cruzeiro do Sul Virtual
Educação a Distância

Para saber mais, leia os capítulos iniciais dos e-books:

PERKOVIC, Ljubomir; VIEIRA, Daniel. Introdução à computação usando Python: um foco no desenvolvimento de aplicações. Rio de Janeiro: LTC, 2016.

McKinney, W. Python Para Análise de Dados: Tratamento de Dados com Pandas, NumPy e IPython. São Paulo: Novatec, 2018