

# VISUALIZAÇÃO COMPUTACIONAL

**Planejamento da visualização  
e exemplos**

# TÓPICOS

- Planejamento da visualização
- Exemplo

# COMO APRESENTAR DADOS?



Fonte: Stage Curtains, Anne Contet, Flickr

# COMO APRESENTAR DADOS?

Problema?

Público-alvo?

Volume e formato dos dados

O que será feito?

Fonte: Stage Curtains, Anne Contet, Flickr

# COMO APRESENTAR DADOS?

Muitas visualizações são ineficazes:

- Diversas possibilidades para cada contexto específico
- Atender à percepção e cognição humana
- Atender aos objetivos da visualização

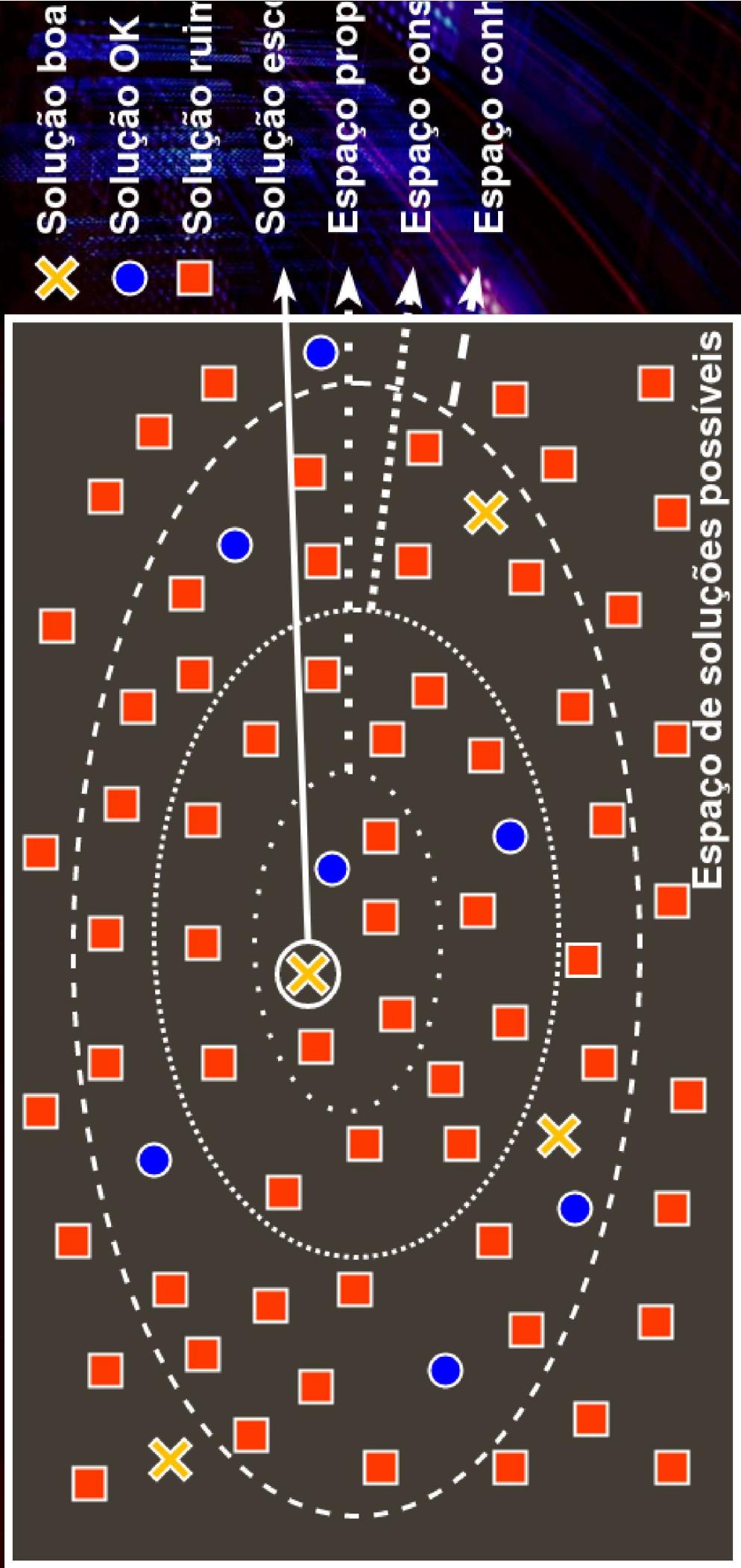
# COMO APRESENTAR DADOS?

**Muitas visualizações são ineficazes:**

- Diversas possibilidades para cada contexto específico
- Atender à percepção e cognição humana
- Atender aos objetivos da visualização

Planejar a visualização ajuda a escolher boas opções para um determinado domínio, de acordo com as características dos dados.

# COMO APRESENTAR DADOS?



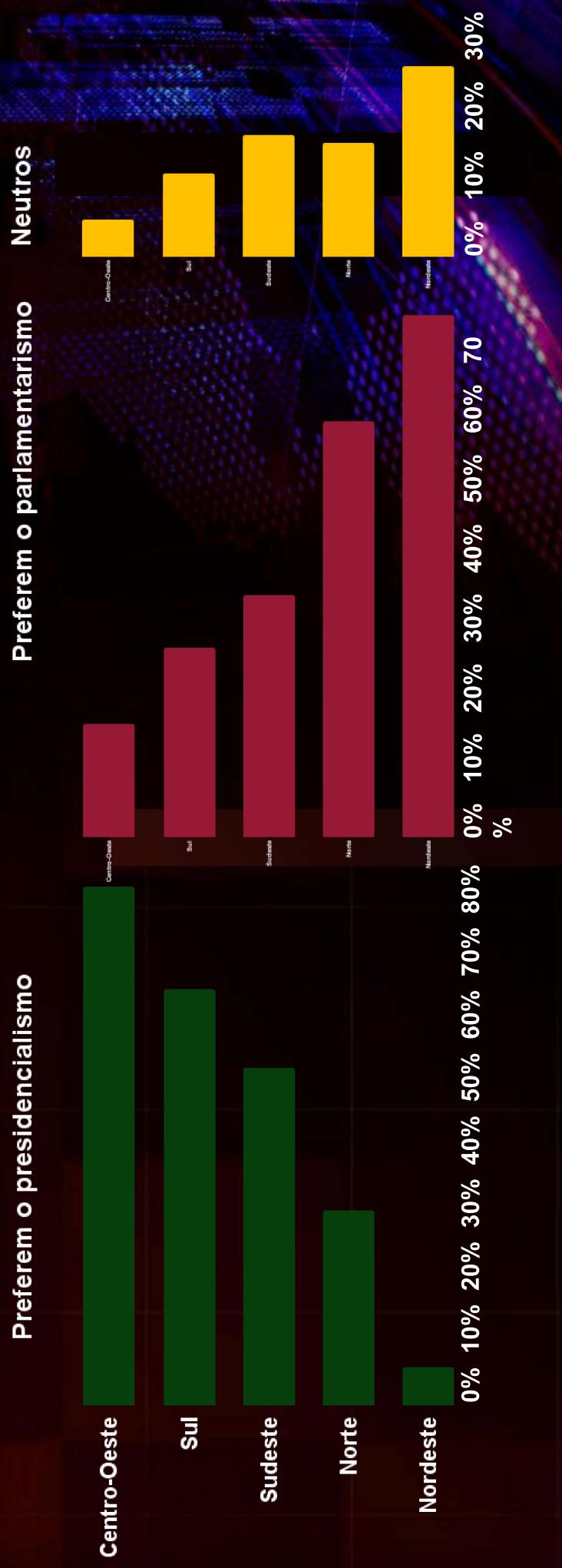
Fonte: Munzner, 2014

# COMO APRESENTAR DADOS? SOLUÇÃO RU

<b>Opinião sobre o regime de governo</b>	
Centro-Oeste: preferem o presidencialismo	80%
Centro-Oeste: preferem o parlamentarismo	15%
<b>Nordeste: preferem o presidencialismo</b>	<b>6%</b>
<b>Nordeste: preferem o parlamentarismo</b>	<b>69%</b>
Norte: preferem o presidencialismo	30%
Norte: preferem o parlamentarismo	55%
<b>Sudeste: preferem o presidencialismo</b>	<b>52%</b>
<b>Sudeste: preferem o parlamentarismo</b>	<b>32%</b>
Sul: preferem o presidencialismo	64%
Sul: preferem o parlamentarismo	25%

Baseado em: Few, 2012

# COMO APRESENTAR DADOS? SOLUÇÃO BO



Baseado em: Few, 2012

Opinião sobre o regime de governo no Brasil por região

# PLANEJAMENTO DA VISUALIZAÇÃO

**Modelo aninhado de quatro níveis - Munzner (2009)**

# PLANEJAMENTO DA VISUALIZAÇÃO

## Modelo aninhado de quatro níveis - Munzner (2009)

- Quatro níveis de planejamento
- Abordar cada nível separadamente
- Pode ser combinado com outras estratégias

# MODELO DE MUNZNER



Fonte: Milani et al., 2020

# MODELO DE MUNZNER - DOMÍNIO

Domínio

Abstração

Idioma

Algoritmo

- Área do domínio:
- Diversas possibilidades
- Usuário final:
- Conhecimento prévio dos dados
- Técnicas de visualização conhecidas
- Público-alvo:
- Perfil estratégico ou informativo
- Perfil investigativo ou exploratório

# PÚBLICO-ALVO

- Perfil estratégico ou informativo:

- Dados sumarizados
- Uso de indicadores
- Tomada de decisão

- Visualizações mais tradicionais:

- Gráficos de barra, pizza, linhas...
- Painéis (dashboards)

Domínio

Abstração

Idioma

Algoritmo

# PÚBLICO-ALVO – PERFIL ESTRATÉGICO

**Fonte:** Wiko BauSoftware, Wikimedia Commons

# PÚBLICO-ALVO

Domínio

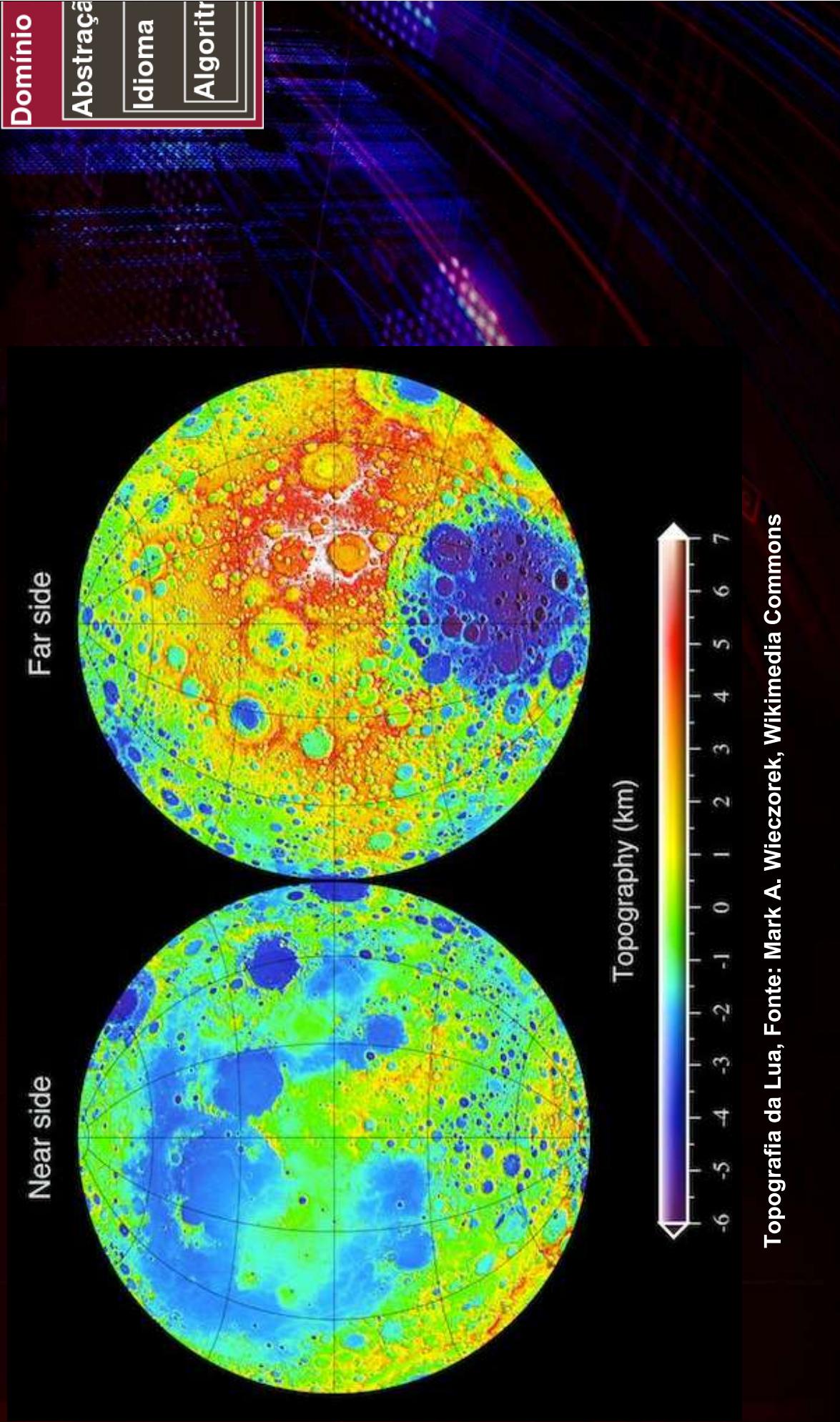
Abstração

Idioma

Algoritmo

- Perfil investigativo ou exploratório:
- Cientistas e analistas
- Dados detalhados
- Descoberta de conhecimento
- Diferentes níveis de informação
- Visualizações não convencionais
- Múltiplas fontes de dados

# PÚBLICO-ALVO – PERFIL INVESTIGATIVO



Domínio  
Abstração  
Idioma  
Algoritmo

# MODELO DE MUNZNER - ABSTRAÇÃO

- Independente do domínio
- Abstração de tarefas:
  - Navegar, comparar ou resumir
  - Abstração de dados:
    - Transformação para formatos específicos
    - Atributos, links, posição, grids...
  - Dimensões

Dominio

Abstração

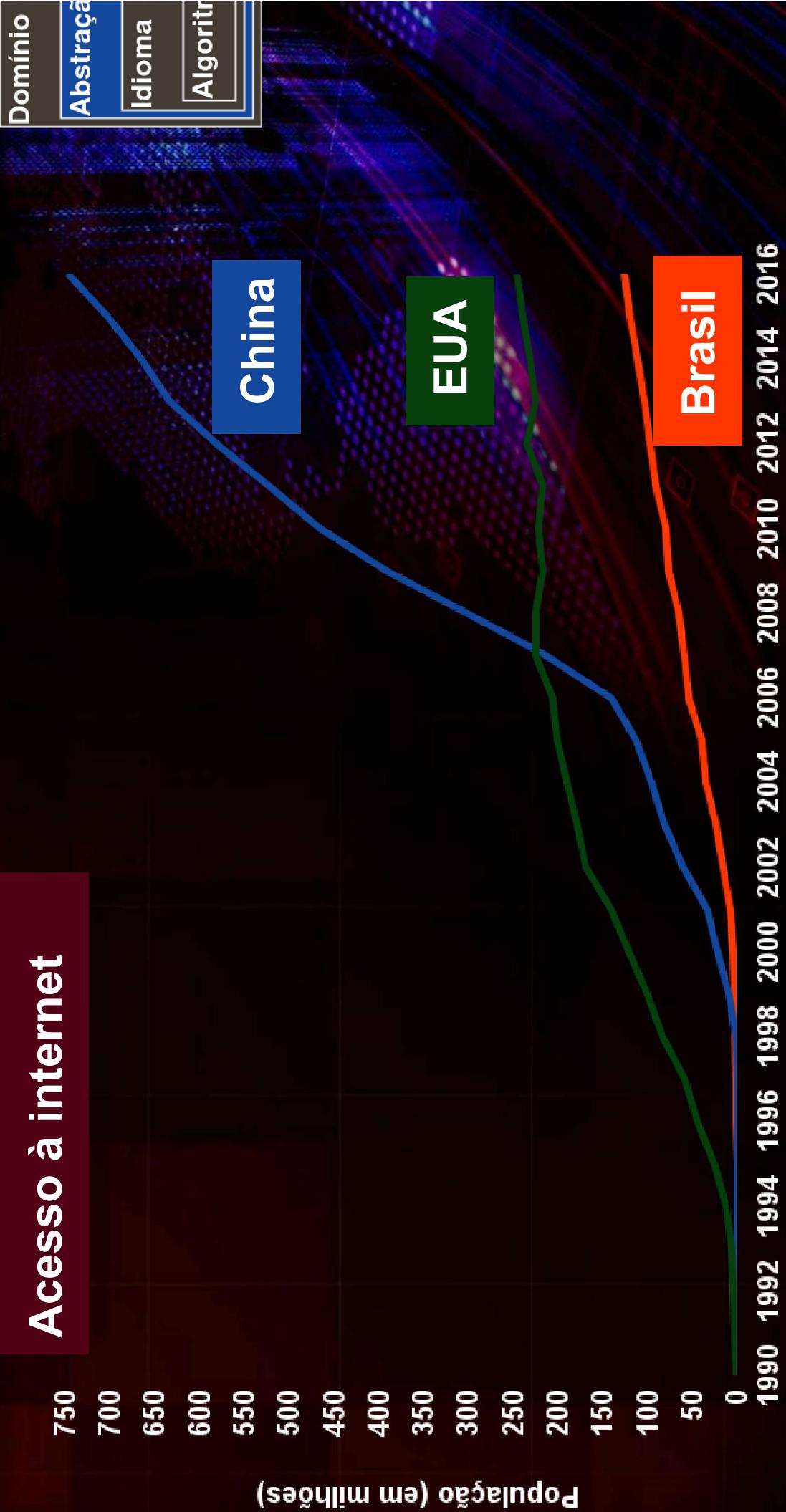
Idioma

Algoritmo

# ABSTRAÇÃO

Acesso à internet

Fonte: Our World in Data



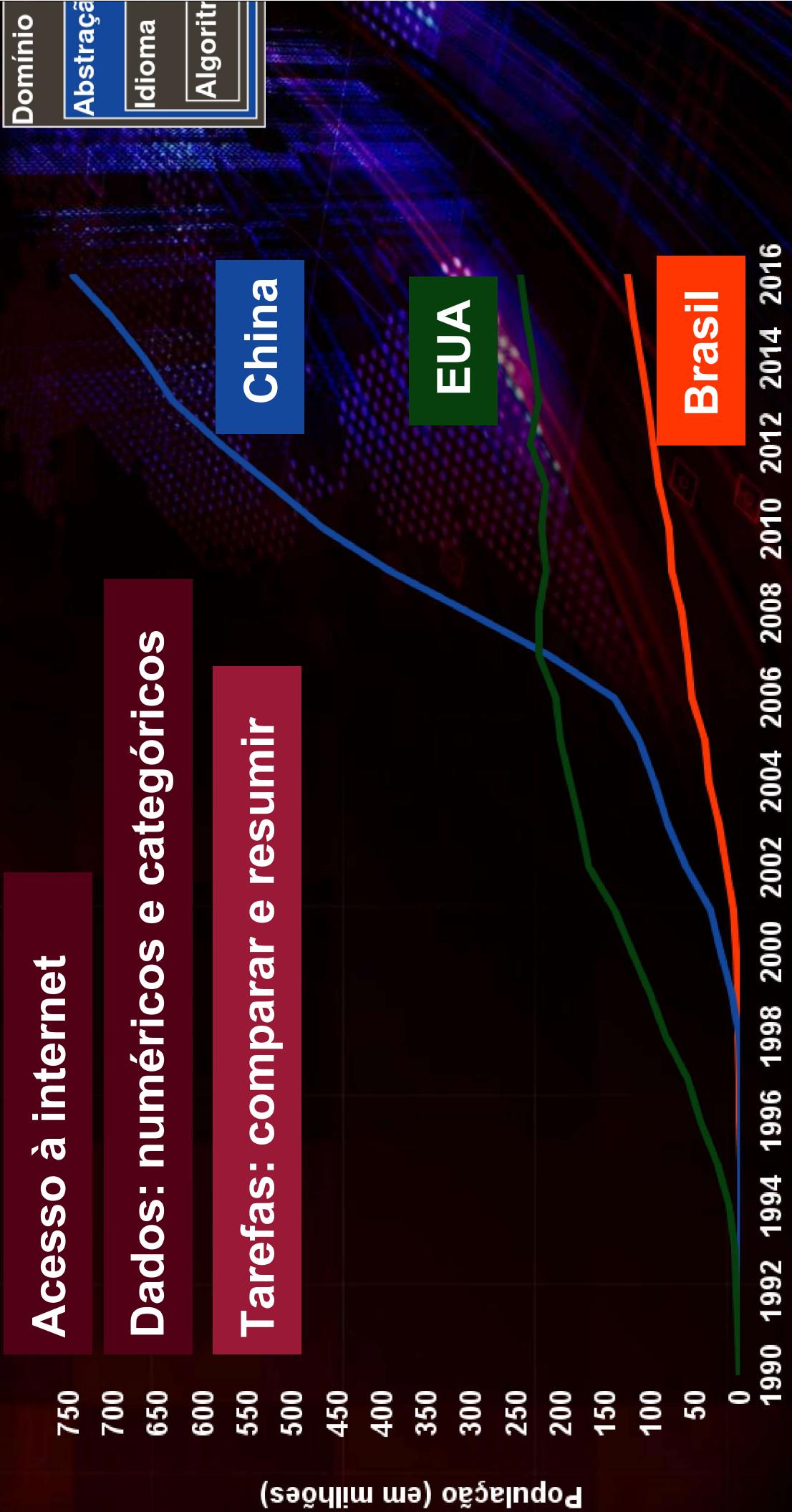
# ABSTRAÇÃO

Acesso à internet

Dados: numéricos e categóricos

Tarefas: comparar e resumir

Fonte: Our World in Data



Domínio  
Abstração  
Idioma  
Algoritmo

# MODELO DE MUNZNER – IDIOMA



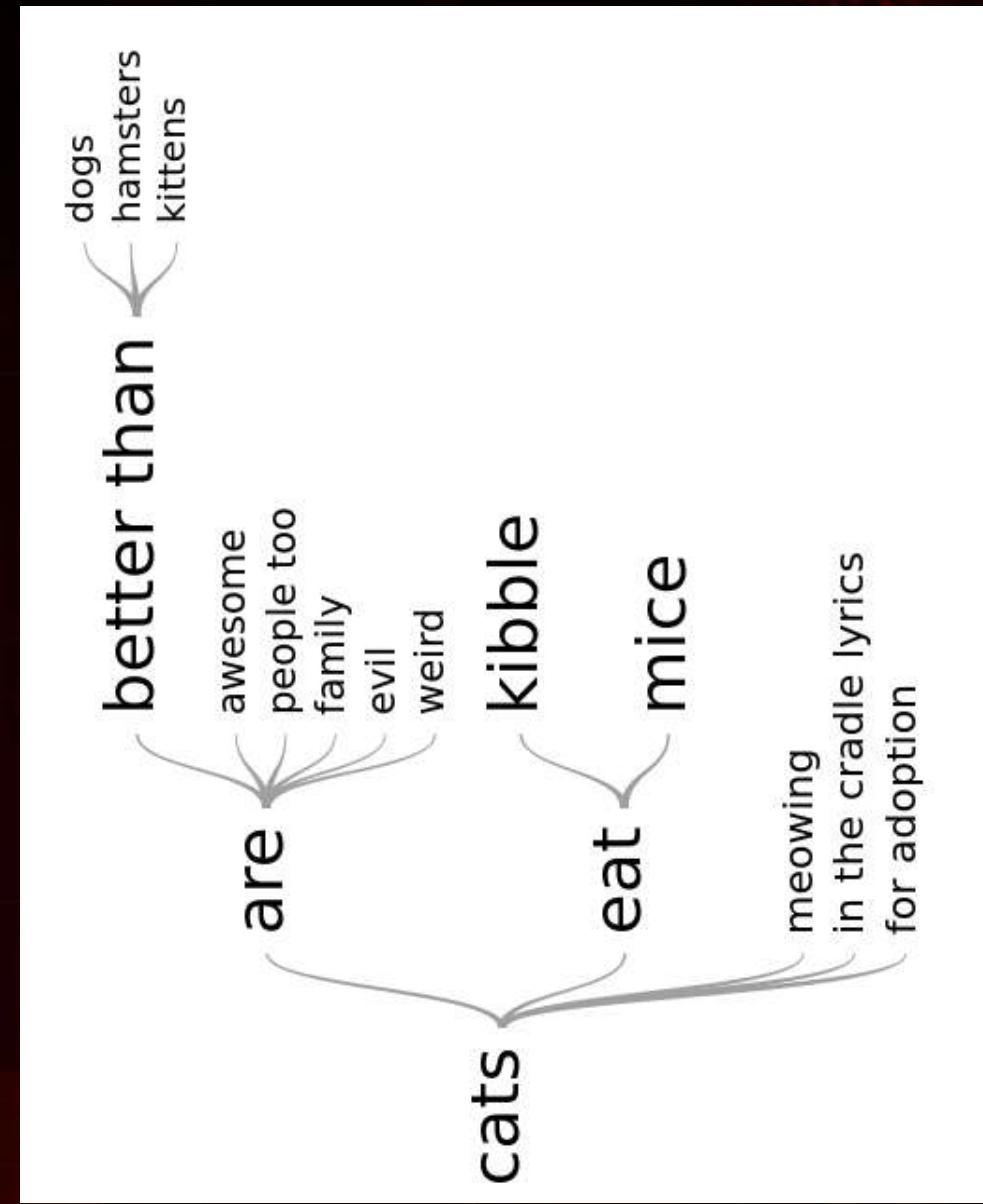
- **Idioma:**
  - Forma específica de criar e manipular a visualização
- **Codificação visual:**
  - O que os usuários veem
- **Interação:**
  - Como os usuários mudam o que veem

# MODELO DE MUNZNER – IDIOMA



- Uso de múltiplos idiomas:
  - Usuários escolhem a visualização mais adequada

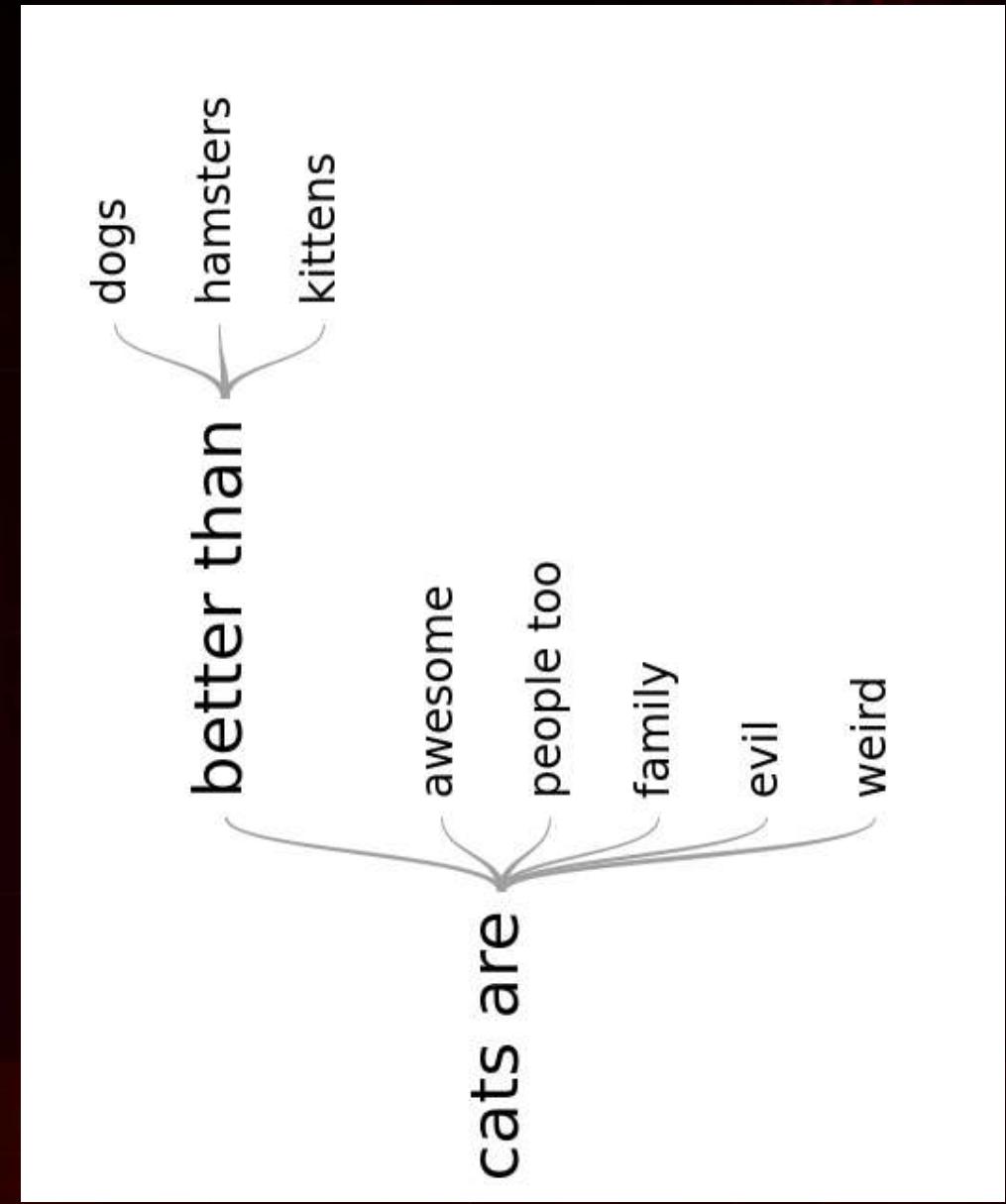
# IDIOMA - ÁRVORE DE PALAVRAS



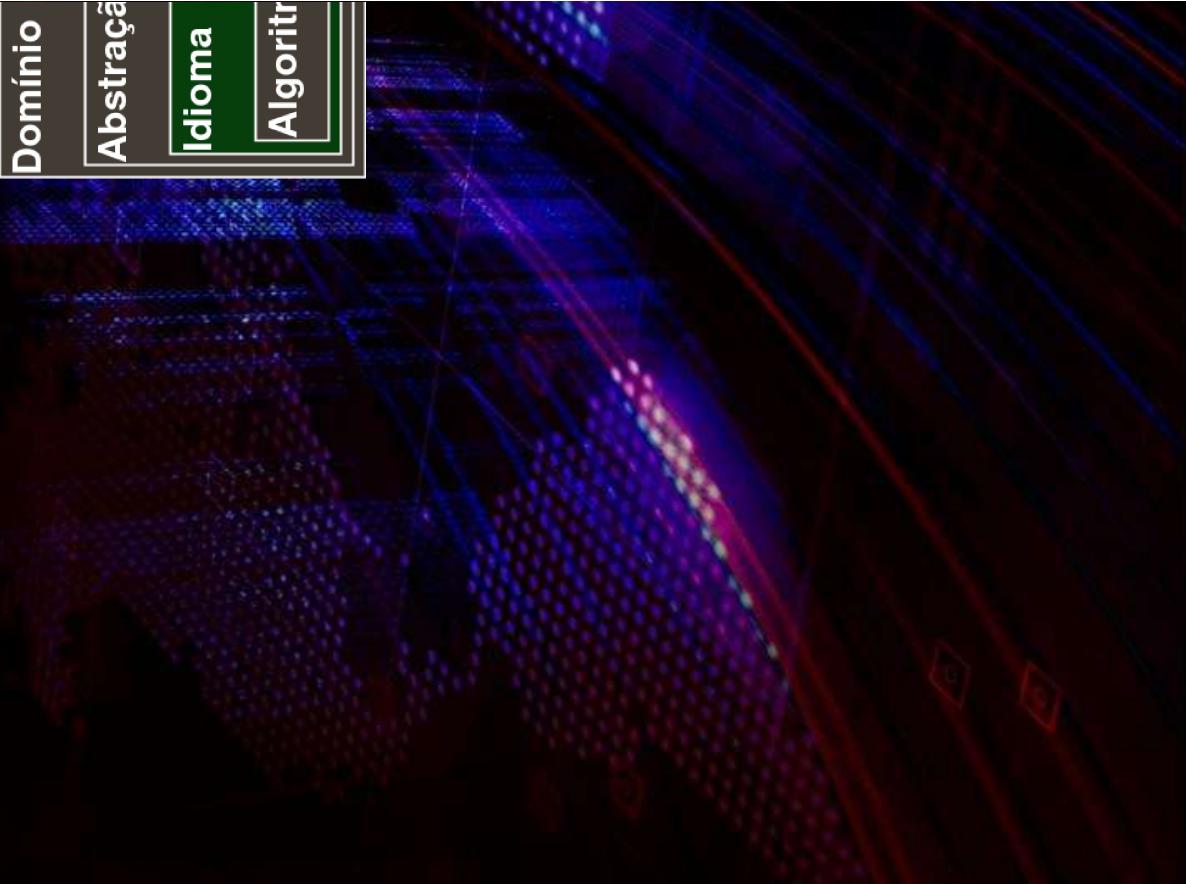
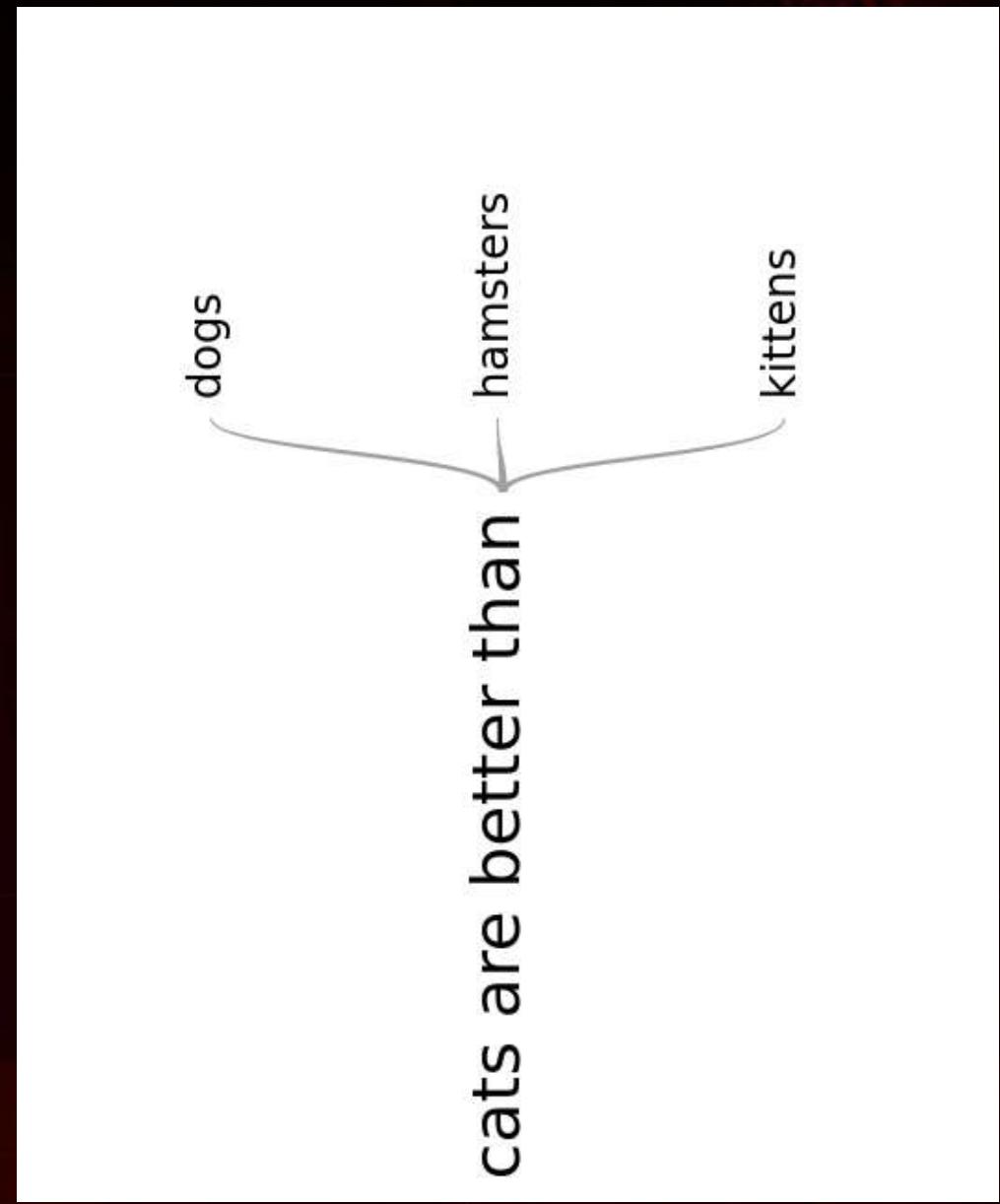
Fonte: Google Charts

Domínio  
Abstração  
Idioma  
Algoritmo

# IDIOMA - ÁRVORE DE PALAVRAS



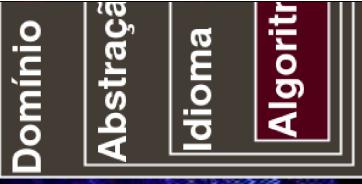
# IDIOMA - ÁRVORE DE PALAVRAS



Domínio  
Abstração  
Idioma  
Algoritmo

Fonte: Google Charts

# MODELO DE MUNZNER – ALGORITMO



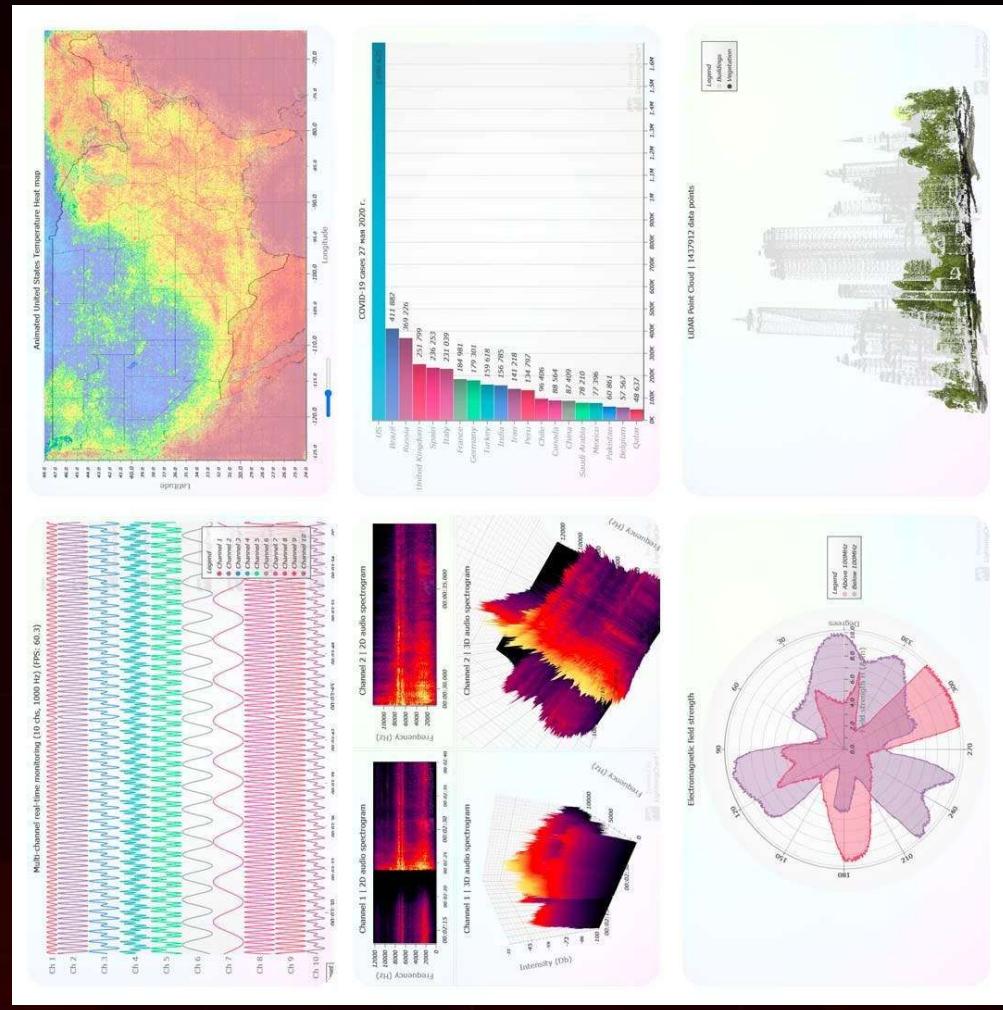
- **Procedimento detalhado da visualização**
- **Automatizar a criação da visualização**
- **Pode haver diferentes algoritmos:**
  - **Extração / transformação de dados**
  - **Renderização**
  - **Publicação do idioma**

# MODELO DE MUNZNER – ALGORITMO

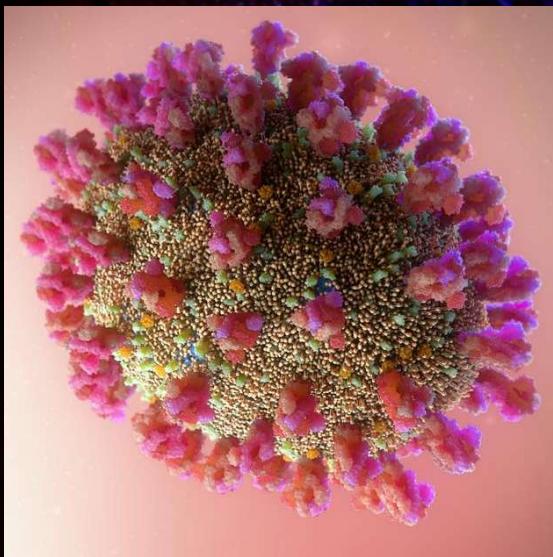


- Mais comum em visualizações complexas
- Criação ou reúso de algoritmos existentes
- Avaliação de desempenho e validação

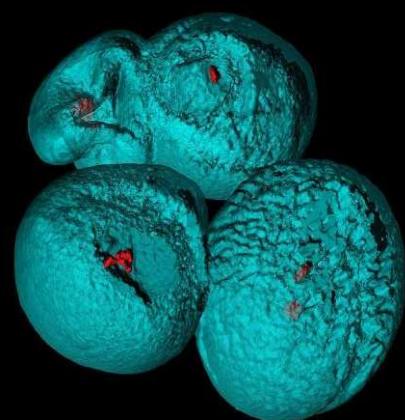
# EXEMPLOS



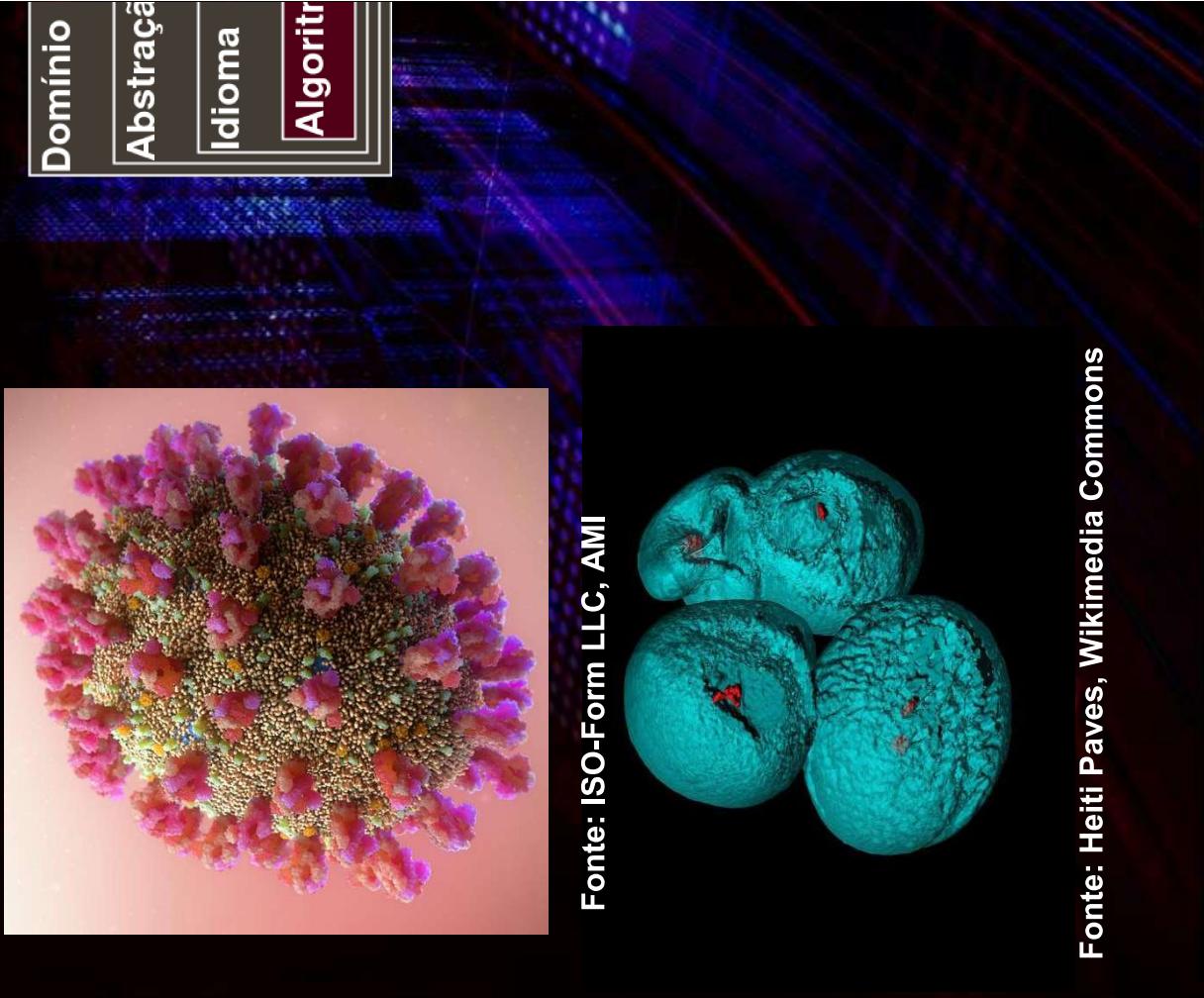
Fonte: Dmitrii Rabtsevich, Lightning Chart JS



Fonte: ISO-Form LLC, AMI



Fonte: Heiti Paves, Wikimedia Commons



**Domínio**  
**Abstração**  
**Idioma**  
**Algoritmo**

# PLANEJAMENTO DA VISUALIZAÇÃO

Tipologia multinível de abstração de tarefas Brehmer  
e Munzner (2013)

# PLANEJAMENTO DA VISUALIZAÇÃO

**Tipologia multinível de abstração de tarefas Brehmer e Munzner (2013)**

- Definir tarefas específicas para criar visualizações
- Responder aos objetivos da visualização:
  - O que?
  - Por que?
  - Como?

# PLANEJAMENTO DA VISUALIZAÇÃO

## Domínio

Caracterizar o problema

Qual a necessidade? Quem é o público-alvo?

## Abstração

O que deve ser apresentado?

Por que o usuário estará olhando?

O que?

Por que?

## Idioma

Como será apresentado?

Como?

## Algoritmo

A computação é eficiente?

Fonte: Milani et al., 2020

# PLANEJAMENTO DA VISUALIZAÇÃO

## Domínio

Caracterizar o problema

Qual a necessidade? Quem é o público-alvo?

## Abstração

O que deve ser apresentado?

Por que o usuário estará olhando?

## Idioma

Como será apresentado?

## Algoritmo

A computação é eficiente?

- Conjunto de dados
  - Tipos de dados
  - Tipos de dataset
  - Disponibilidade
- Atributos
  - Tipo dos atributos
  - Direção

# PLANEJAMENTO DA VISUALIZAÇÃO

## Domínio

Caracterizar o problema

Qual a necessidade? Quem é o público-alvo?

## Abstração

O que deve ser apresentado?

Por que o usuário estará olhando?

O que?

Por que?

## Idioma

Como será apresentado?

## Algoritmo

A computação é eficiente?

- Ações
  - Analisar (consumo; produção)
  - Pesquisar
  - Consultar (identificar; comparar)
- Objetivos
  - Visão do todo (tendências; outliers)
  - Atributos (único; vários)

# PLANEJAMENTO DA VISUALIZAÇÃO

## Domínio

Caracterizar o problema

**Qual a necessidade? Quem é o público-alvo?**

## Abstração

O que deve ser apresentado?

Por que o usuário estará olhando?

## Idioma

Como será apresentado?

## Algoritmo

A computação é eficiente?

## Codificar

- Organizar; mapear (cor; tamanho)
- Manipular
  - Alterar; selecionar; navegar
- Facetar
  - Particionar; sobrepor
- Reduzir
  - Filtrar; agrregar; embutir

**O que?**

**Por que?**

**Como?**

# TÓPICOS

1. Planejamento da visualização

2. Exemplo

# JAGUAR - Java coveraGe faUlt locAlization R

- Localização de defeitos automatizada
- Usa informações de teste automatizado
- Gera uma lista de códigos suspeitos de conter defeitos
- Plug-in do Eclipse ou linha de comando
- Código aberto: <https://github.com/saeg/jaguar>

# JAGUAR - DOMÍNIO

- Necessidade:
  - Automatizar a localização de defeitos
- Público-alvo:
  - Desenvolvedores e testadores

# JAGUAR - ABSTRAÇÃO

- O que deve ser apresentado?
  - Código
  - Resultado de testes
- Por que o usuário estará olhando?
  - Encontrar defeitos do programa

# JAGUAR - IDIOMA

- Como será apresentado?
- Plug-in do IDE Eclipse
- Lista de códigos mais suspeitos
- Uso de cores de acordo com o nível de suspeição
- Diferentes granularidades: métodos ou linhas
- Navegar pelo código a partir da lista
- Filtrar e buscar

# JAGUAR - ALGORITMO

- Eficiência (projetos grandes):
  - Lista é gerada em segundos
  - Cálculo do código suspeito pode levar alguns minutos
- Procedimento:
  - Rodar testes
  - Calcular suspeição
  - Gerar lista
  - Exibir no Eclipse

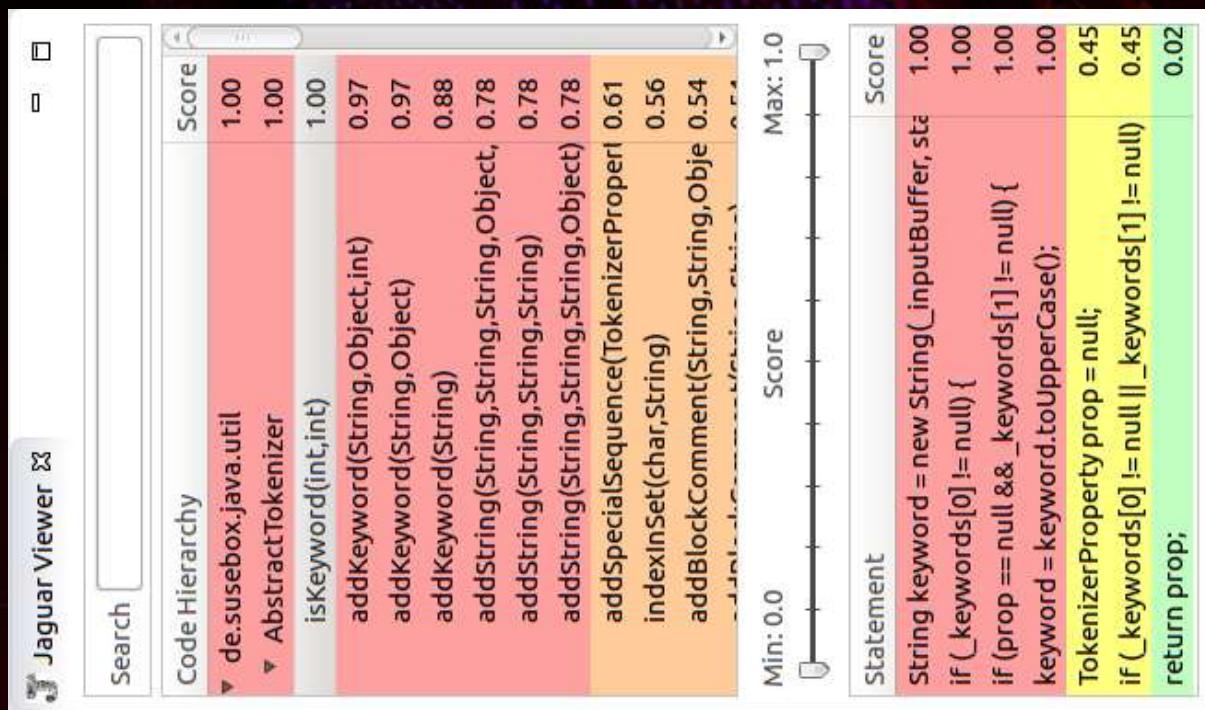
# JAGUAR

The screenshot shows the JAGUAR IDE interface with the following components:

- File Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Toolbars:** Standard Java development tools like Open, Save, Cut, Copy, Paste, Find, etc.
- Project Explorer:** Shows the project structure with files like `AbstractTokenizer.java`, `CalcExample.java`, `hsqldb`, `jsoup`, `jtopas-0.4`, `junit`, `src`, `JRE System`, `JUnit 4`, `jaguar.xml`, and `xstream`.
- Code Editor:** The `AbstractTokenizer.java` file is open, showing Java code with annotations and code analysis results. Annotations include:
  - Line 1552: `length` annotated with `param`.
  - Line 1553: A note: `return @Link TokenizerProperty} describing the keyword or <code>`.
  - Line 1554: `protected TokenizerProperty isKeyword(int,int)` annotated with `Class.Method` and a score of 1.00.
  - Line 1555: `TokenizerProperty prop = null;` annotated with `AbstractTokenizer.addKeyword(Stri`.
  - Line 1556: `// test on keyword` annotated with `AbstractTokenizer.addKeyword(Stri`.
  - Line 1558: `if (-keywords[0] != null || -keywords[1] != null) {` annotated with `AbstractTokenizer.addKeyword(Stri`.
  - Line 1559: `String keyword = new String( inputBuffer, startingAtPos - range` annotated with `AbstractTokenizer.addString(String,`.
  - Line 1560: `if (-keywords[0] != null) {` annotated with `AbstractTokenizer.addString(String,`.
  - Line 1561: `prop = (TokenizerProperty)-keywords[0].get(keyword);` annotated with `AbstractTokenizer.addString(String,`.
  - Line 1563: `if (prop == null && -keywords[1] != null) {` annotated with `AbstractTokenizer.addSpecialSequence(Tc`.
  - Line 1564: `keyword = keyword.toUpperCase();` annotated with `AbstractTokenizer.addSpecialSequence(Tc`.
  - Line 1565: `prop = (TokenizerProperty)-keywords[0].get(keyword);` annotated with `AbstractTokenizer.addSpecialSequence(Tc`.
  - Line 1566: `}` annotated with `AbstractTokenizer.findIndexSet(char,int[])`.
  - Line 1568: `return prop;` annotated with `AbstractTokenizer.findIndexSet(char,int[])`.
  - Line 1570: `}` annotated with `AbstractTokenizer.findIndexSet(char,int[])`.
  - Line 1572: `// Implementation` annotated with `AbstractTokenizer.addBlockComm`.
  - Line 1574: `//` annotated with `AbstractTokenizer.addBlockComm`.
  - Line 1576: `//` annotated with `AbstractTokenizer.addBlockComm`.
  - Line 1577: `/**` annotated with `AbstractTokenizer.isSequenceComm`.
  - Line 1578: `* This method creates the sorted arrays to store the case-sensitive` annotated with `InputStreamTokenizer.read(char[],int`.
  - Line 1579: `* -insensitive special sequences, comments, strings etc.` annotated with `0.52`.
  - Line 1580: `*` annotated with `AbstractTokenizer.read(char[],int`.
  - Line 1581: `* @param prop the description of the new sequence` annotated with `0.53`.
  - Line 1582: `*/` annotated with `AbstractTokenizer.read(char[],int`.
  - Line 1583: `protected void addSpecialSequence(TokenizerProperty prop) {` annotated with `0.53`.
  - Line 1584: `int arrayIdx;` annotated with `AbstractTokenizer.getParseFlags()`.
  - Line 1585: `int flags = prop.getFlags();` annotated with `0.52`.
  - Line 1586: `}` annotated with `AbstractTokenizer.getParseFlags()`.
- Search Bar:** Search: [ ]
- Analysis Results:** Shows a table of analysis results for methods:

Method	Score
AbstractTokenizer.isKeyword(int,int)	1.00
AbstractTokenizer.addKeyword(String, int)	0.97
AbstractTokenizer.addKeyword(String, int)	0.97
AbstractTokenizer.addString(String, int)	0.88
AbstractTokenizer.addString(String, int)	0.78
AbstractTokenizer.addString(String, int)	0.78
AbstractTokenizer.addString(String, int)	0.78
AbstractTokenizer.addSpecialSequence(Token, int)	0.65
AbstractTokenizer.addSpecialSequence(Token, int)	0.61
AbstractTokenizer.findIndexSet(char, int[])	0.61
AbstractTokenizer.findIndexSet(char, int[])	0.59
AbstractTokenizer.findIndexSet(char, int[])	0.56
AbstractTokenizer.addBlockComment(int, int)	0.54
AbstractTokenizer.addBlockComment(int, int)	0.54
AbstractTokenizer.isSequenceComment(int, int)	0.53
InputStreamTokenizer.read(char[], int)	0.53
AbstractTokenizer.getParseFlags()	0.52
- Metrics:** Min: 0.0, Current min score: 0.0, Max: 1.0.
- Failure Trace:** Shows a failure trace with 0 errors, 0 failures, and 1556 inserts.
- Bottom Bar:** Tasks, Console, JUnit, Writable, Smart Insert, and a status bar showing Runs: 0/0.

# JAGUAR



# REFERÊNCIAS

**Visualização de dados. Capítulo: Visualização de dados em big data.**  
**Alessandra M. Paz Milani et al. Editora SAGA-H, 2020.**

**Visualization Analysis and Design.** Tamara Munzner. CRC Express, 2014.

**Show Me The Numbers – Designing Tables and Graphs to Enlighten, 2a. edição.**  
Stephen Few. Analytics Press, 2012.

**Google Charts – Wordtree**

Link: <https://developers.google.com/chart/interactive/docs/gallery/wordtree>

**Automatização de Teste de Software com Ferramentas de Software Livre. Cap. 9 - Ferramentas de Depuração de Software. Auri M. R. Vincenzi et al. GEN LTC, 2018.**

**Repositório da Jaguar:** <https://github.com/saeg/jaguar>

# VISUALIZAÇÃO COMPUTACIONAL

**Planejamento da visualização  
e exemplos**