

Visualización para ciencia de datos

Framework visualización

¿Qué? ¿Por qué? ¿Cómo?

Contenido

1

¿Qué?

¿Qué datos ve el usuario?

2

¿Por qué?

¿Por qué el usuario tiene la intención de utilizar una herramienta de vis?

3

¿Cómo?

¿Cómo la codificación visual y la interacción los modismos se construyen en términos de opciones de diseño?

¿Qué es un framework de visualización?

Es un marco de análisis que impone una estructura en este enorme espacio de diseño, pensado como un andamio para ayudarte a pensar en las opciones de diseño de forma sistemática.

Framework **What**, **Why**, **How**?

Logra una metodología para lograr **abstracciones** efectivas que cumplan las tareas que el usuario quiere ver.

¿Hay más? Sí

¿Es una camisa de fuerza? No

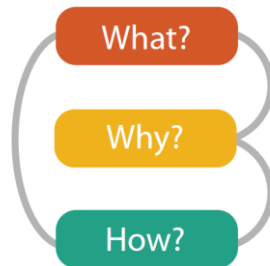
¿Por qué éste?



Porque soy Tamara Munzner ¿Ud no sabe quién soy yo?



Ben Shneiderman
PhD Universidad de Maryland



¿Qué?

Abstracción de los datos

¿Qué datos ve el usuario?

¿Qué tipo de datos se proporcionan?

¿Qué información puedes deducir de los datos, frente a los significados que te deben decir explícitamente?

¿Qué conceptos de alto nivel te permitirán dividir los conjuntos de datos en piezas generales y útiles?

¿Qué datos hay aquí?

Atributos/Variables

Items	ID	Name	Age	Shirt Size	Favorite Fruit	Tabla	Estático	Valor
	1	Amy	8	S	Apple			
	2	Basil	7	S	Pear			
	3	Clara	9	M	Durian			
	4	Desmond	13	L	Elderberry			
	5	Ernest	12	L	Peach			
	6	Fanny	10	S	Lychee			
	7	George	9	M	Orange			
	8	Hector	8	L	Loquat			
	9	Ida	10	M	Pear			
	10	Amy	12	M	Orange			

Secuencial

Categoría

What?

Datasets

Attributes

→ Data Types

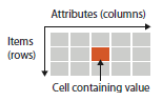
→ Items → Attributes → Links → Positions → Grids

→ Data and Dataset Types

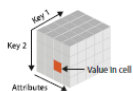
Tables	Networks & Trees	Fields	Geometry	Clusters, Sets, Lists
Items	Items (nodes)	Grids	Items	Items
Attributes	Links	Positions	Positions	
	Attributes	Attributes		

→ Dataset Types

→ Tables



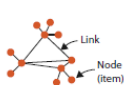
→ Multidimensional Table



→ Geometry (Spatial)



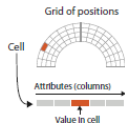
→ Networks



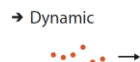
→ Trees



→ Fields (Continuous)



→ Dataset Availability



→ Attribute Types

→ Categorical



→ Ordered

→ Ordinal



→ Quantitative



→ Ordering Direction

→ Sequential



→ Diverging



→ Cyclic



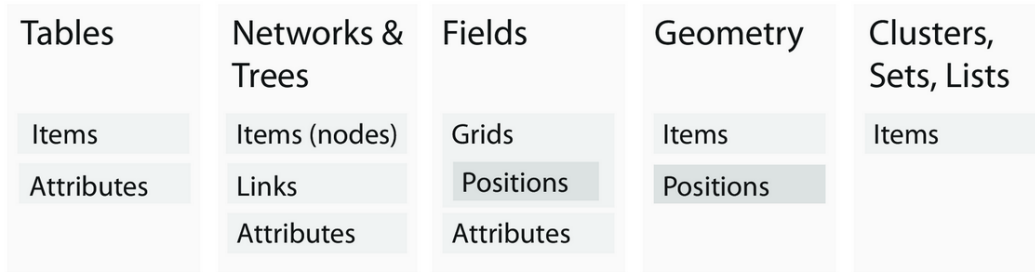
What?

Why?

How?

Tipos de datos, dataset y disponibilidad

➔ Data and Dataset Types



- Temporal

- Items
- Atributos
- Tiempo (seg, min, hora, día, mes, año, día sol en Marte)

➔ Data Types

➔ Items ➔ Attributes ➔ Links ➔ Positions ➔ Grids

➔ Dataset Availability

➔ Static



➔ Dynamic



Tipos de atributos

➔ Attribute Types

➔ Categorical

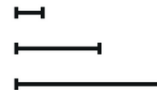


➔ Ordered

➔ Ordinal



➔ Quantitative



➔ Ordering Direction

➔ Sequential



➔ Diverging



➔ Cyclic



¿Por qué?

Abstracción de las tareas

¿Qué tipo de tarea requiere el usuario?

¿Requiere el usuario realizar transformaciones en los datos?

¿Existen datos particulares que requiere el usuario ver?

Why?

Actions

Analyze

→ Consume

→ Discover



→ Present



→ Enjoy



→ Produce

→ Annotate



→ Record



→ Derive



Search

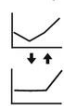
	Target known	Target unknown
Location known	• • • Lookup	• • • Browse
Location unknown	• • • Locate	• • • Explore

Query

→ Identify



→ Compare



→ Summarize



Targets

All Data

→ Trends



→ Outliers



→ Features



Attributes

→ One

→ Distribution



→ Extremes



→ Many

→ Dependency



→ Correlation



→ Similarity



Network Data

→ Topology



→ Paths



Spatial Data

→ Shape



Un conjunto que tiene **verbos** que describen acciones, y **sustantivos** que describen objetivos

- Descubrir distribuciones
- Derivar atributos
- Localizar datos atípicos
- Comparar tendencias



Acciones de alto nivel: Analizar

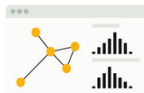
➔ Analyze

➔ Consume

➔ Discover



➔ Present



➔ Enjoy



➔ Produce

➔ Annotate



➔ Record



➔ Derive



Consumir

- Descubrir(explorar) vs presentar (explicar)
- Disfrutar

Producir

- Anotar o recordar
- Derivar ¡Nuevos atributos en los datos!



Original Data



Derived Data

$$\text{trade balance} = \text{exports} - \text{imports}$$

Ejemplo enjoy:

<http://predominant.ly/>

Ejemplo de producción:





<https://www.nytimes.com/interactive/2017/08/09/upshot/game-of-thrones-chart.html>

Ejemplo Derivación:

<https://distill.pub/2018/building-blocks/>

Acciones de nivel medio y bajo (específicas)

→ Search

	Target known	Target unknown
Location known	 <i>Lookup</i>	 <i>Browse</i>
Location unknown	 <i>Locate</i>	 <i>Explore</i>

¿Qué conoce el usuario?

- Objetivo
- Ubicación

→ Query

→ Identify



→ Compare



→ Summarize



¿Uno, algunos o todos?

- Identificar se refiere a un solo objetivo
- Comparar se refiere a múltiples objetivos
- Resumir se refiere al conjunto completo de posibles objetivos.

Objetivos

→ ALL DATA

→ Trends



→ Outliers



→ Features



→ ATTRIBUTES

→ One

→ Distribution



↓ Extremes



→ Many

→ Dependency



→ Correlation

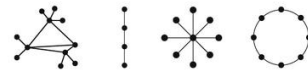


→ Similarity



→ NETWORK DATA

→ Topology



→ Paths



→ SPATIAL DATA

→ Shape



¿Cómo?

Selección del modismo

¿Qué tipo de codificación visual usar?

¿Requiere interacción la visualización?

¿Cuáles características son más efectivas para los tipos de datos?

How?

Encode

➔ Arrange

➔ Express



➔ Order



➔ Use



➔ Separate



➔ Align



➔ Map

from **categorical** and **ordered** attributes

➔ Color

➔ Hue



➔ Saturation



➔ Luminance



➔ Size, Angle, Curvature, ...



➔ Shape



➔ Motion

Direction, Rate, Frequency, ...



Manipulate

➔ Change



➔ Select



➔ Navigate



Facet

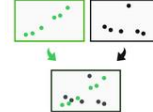
➔ Juxtapose



➔ Partition



➔ Superimpose



Reduce

➔ Filter



➔ Aggregate



➔ Embed

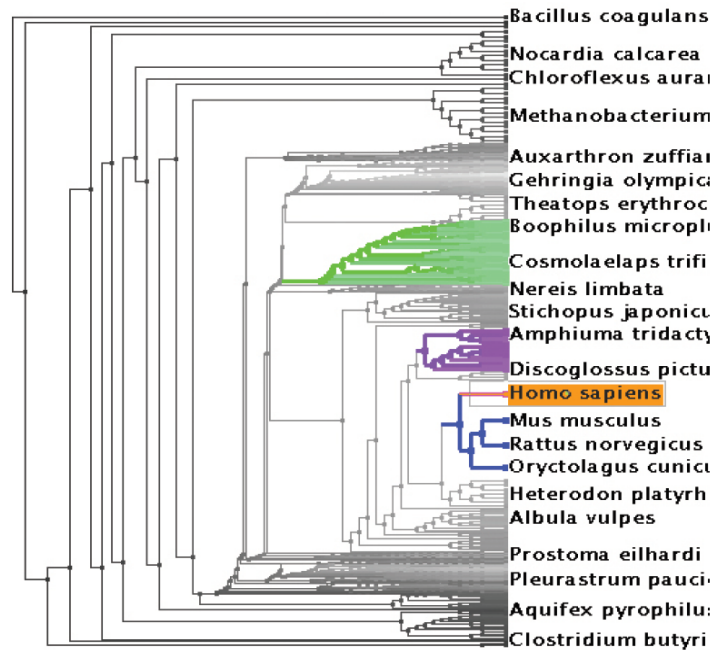
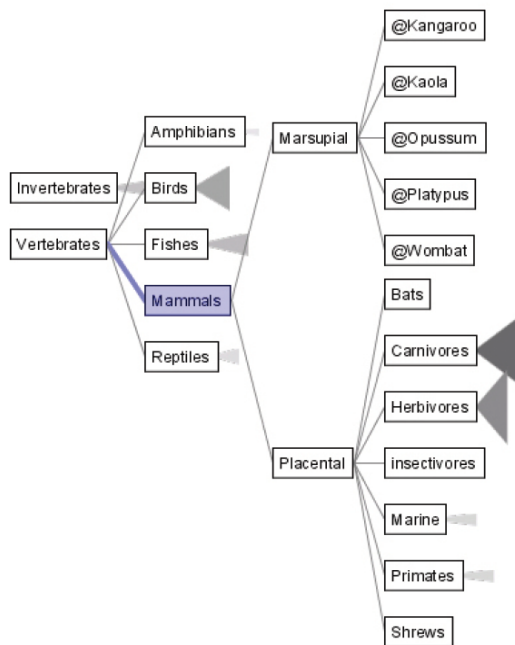


What?

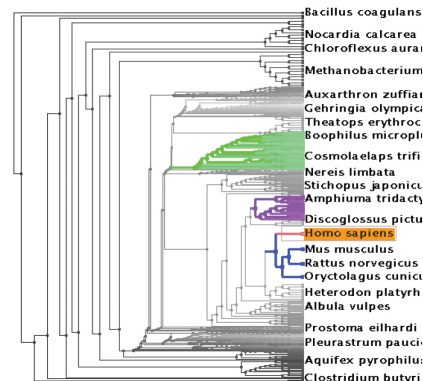
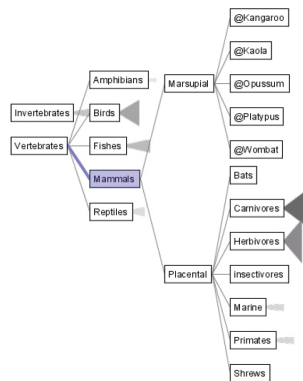
Why?

How?

Ejemplo



Ejemplo



What?

→ Tree



Why?

→ Actions

→ Present → Locate → Identify



→ Targets

→ Path between two nodes



How?

→ SpaceTree

→ Encode → Navigate → Select → Filter → Aggregate



→ TreeJuxtaposer

→ Encode → Navigate → Select → Arrange



Intro a Github

¿Qué es Github?

En un nivel más alto, GitHub es un sitio web y un servicio en la nube que ayuda a los desarrolladores a almacenar y administrar su código, al igual que llevar un registro y control de cualquier cambio sobre este código.



¿Qué es una versión de control?

Una Versión de Control ayuda a los desarrolladores llevar un registro y administrar cualquier cambio en el código del proyecto de software. A medida que crece este proyecto, la versión de control se vuelve esencial.

Con la *bifurcación*, un desarrollador duplica parte del código fuente (llamado *repositorio*). Este desarrollador, luego puede, de forma segura, hacer cambios a esa parte del código, sin afectar al resto del proyecto.

Luego, una vez que el desarrollador logre que su parte del código funcione de forma apropiada, esta persona podría *fusionar* este código al código fuente principal para hacerlo oficial.

¿Qué es Git?

Git es un sistema de control específico de versión de fuente abierta creada por Linus Torvalds en el 2005.

Específicamente, Git es un sistema de control de versión distribuida, lo que quiere decir que la base del código entero y su historial se encuentran disponibles en la computadora de todo desarrollador, lo cual permite un fácil acceso a las bifurcaciones y fusiones.



Fork and clone

La palabra fork se traduce al castellano, dentro del contexto que nos ocupa, como bifurcación. Cuando hacemos un fork de un repositorio, se hace una copia exacta en crudo del repositorio original que podemos utilizar como un repositorio git cualquiera.

Cuando creas un repositorio en GitHub, existe como un repositorio remoto. Puedes clonar tu repositorio para crear una copia local en tu ordenador y sincronizarlo entre las dos ubicaciones.

Commit, push and pull

Ahora que tienes una copia local y una copia en tu cuenta de GitHub, hay cuatro cosas que tendrás que saber hacer para colaborar.

- Commit (Confirmar) es el proceso que registra los cambios en el repositorio. Piensa en ello como una instantánea del estado actual del proyecto. Las confirmaciones se hacen localmente.
- Push (Empujar) envía el historial de confirmaciones recientes de tu repositorio local a GitHub. Si eres el único que trabaja en un repositorio, empujar es bastante simple. Si hay otras personas que acceden al repositorio, es posible que tengas que hacer un pull antes de poder hacer un push.
- Pull - un pull coge cualquier cambio del repositorio de GitHub y lo fusiona en tu repositorio local.
- Sincronización - la sincronización es como halar, pero en lugar de conectarse a su copia de GitHub del repositorio bifurcado, vuelve al repositorio original y trae cualquier cambio. Una vez que hayas sincronizado tu repositorio, tienes que empujar esos cambios de vuelta a tu cuenta de GitHub.

Gracias

¿Preguntas?

Bibliografía

- Munzner Tamara, Visualization Analysis and Design, Department of Computer Science University of British Columbia , Capitulo 2-3 (2014)
- ¿Qué es github? <https://kinsta.com/es/base-de-conocimiento/que-es-github/>
- Fork de repositorios ¿Para qué sirve? <https://aprendegit.com/fork-de-repositorios-para-que-sirve/>
- Clonar un repositorio <https://docs.github.com/es/github/creating-cloning-and-archiving-repositories/cloning-a-repository>
- Basado en el Curso de visualización de datos en D3 por Jhon Alexis Guerra https://johnguerra.co/lectures/visualAnalytics_fall2019/