05. Task Abstraction

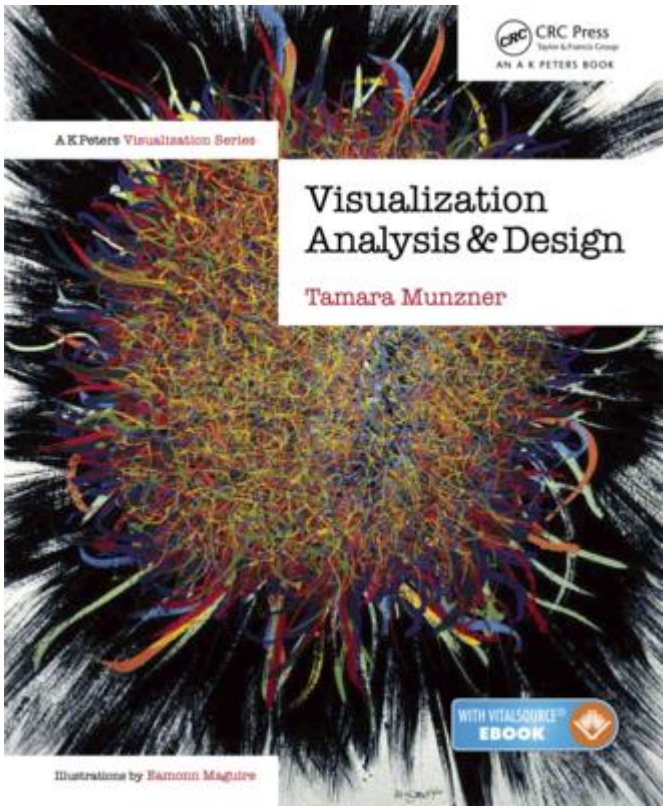
ID 413: Information Graphics and Data Visualization
Spring 2025

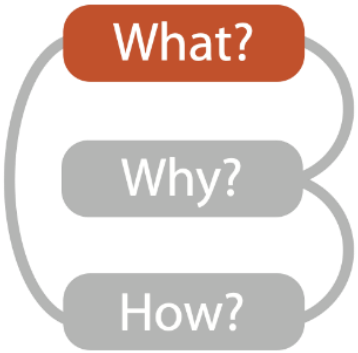
Venkatesh Rajamanickam (@venkatrajam)
venkatra@iitb.ac.in
http://info-design-lab.github.io/

Visualization Analysis & Design

Tamara Munzner

A K Peters Visualization Series
CRC Press, 2014





What?

Datasets

Data Types

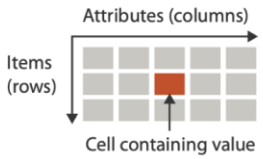
→ Items → Attributes → Links → Positions → Grids

Data and Dataset Types

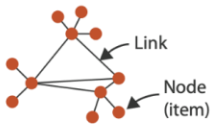
Tables	Networks & Trees	Fields	Geometry	Clusters, Sets, Lists
Items	Items (nodes)	Grids	Items	Items
Attributes	Links	Positions	Positions	
	Attributes	Attributes		

Dataset Types

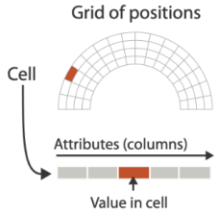
→ Tables



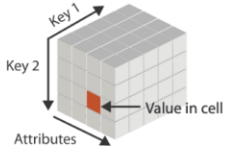
→ Networks



→ Fields (Continuous)



→ *Multidimensional Table*



→ *Trees*

→ Geometry (Spatial)

Dataset Availability

→ Static

→ Dynamic

Attributes

Attribute Types

→ Categorical

→ Ordered

→ *Ordinal*

→ *Quantitative*

Ordering Direction

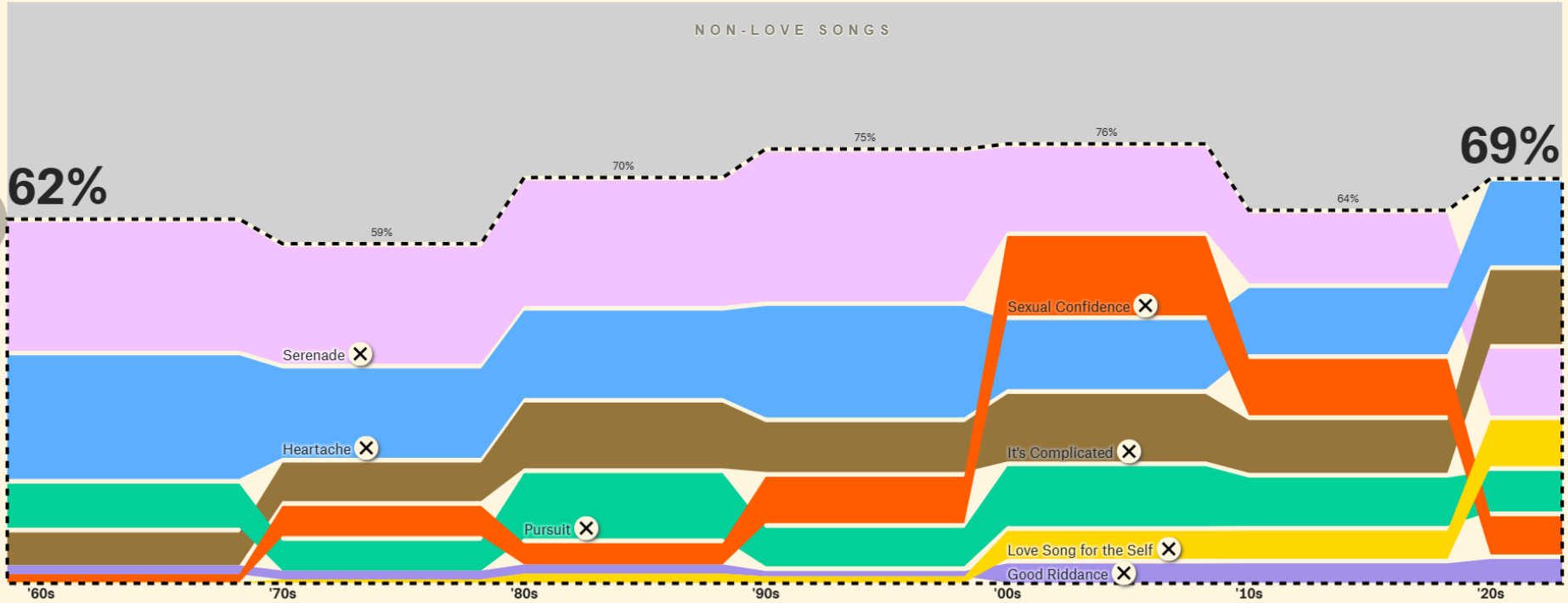
→ Sequential

→ Diverging

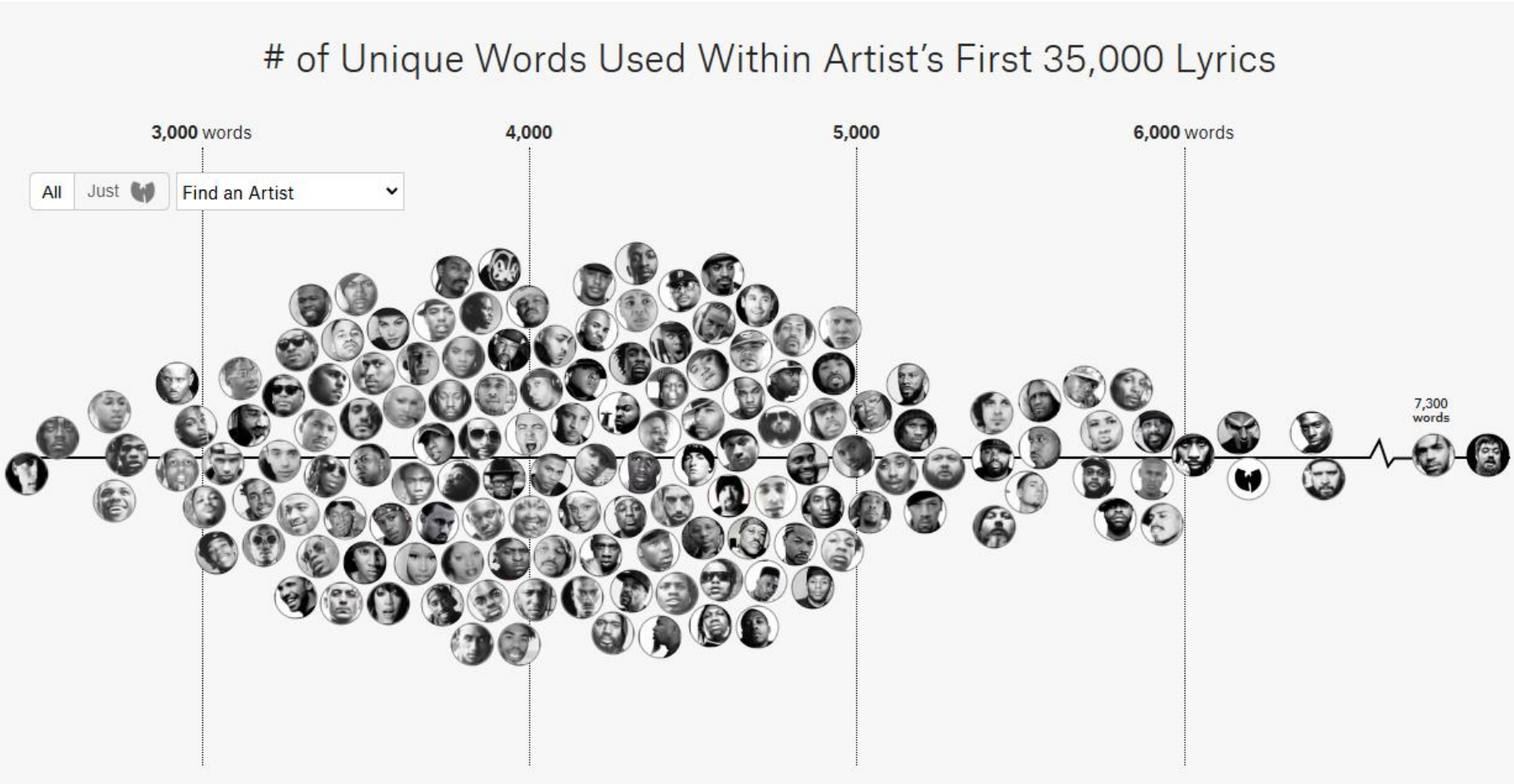
→ Cyclic

That's using our criteria. The status of love songs depends on what *you* believe should or should not count as a love song. So, you decide: **remove categories** ✕ **based on your love song beliefs**, and tell *us* how the love song is doing!

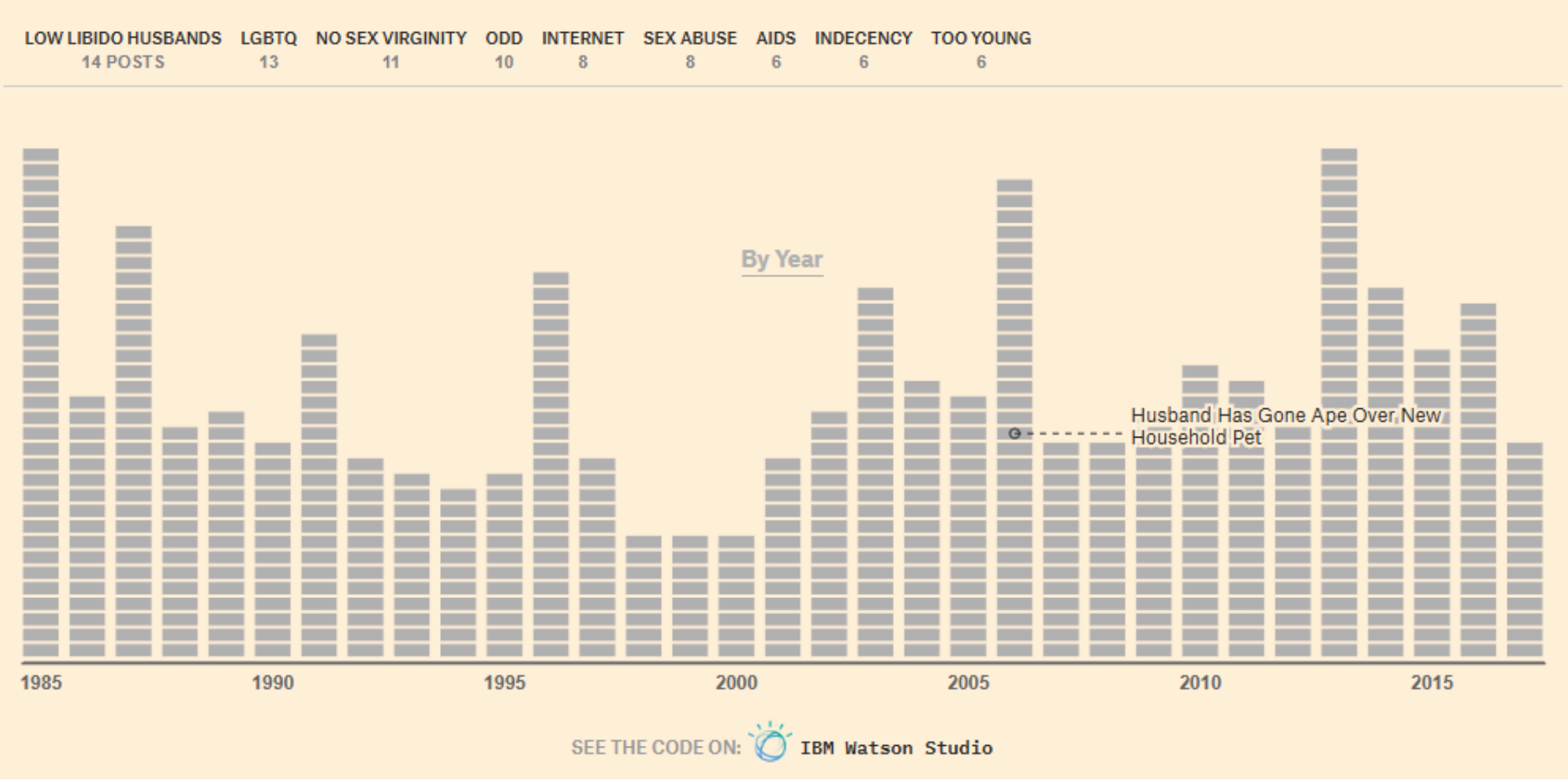
Highlight artists... Highlight songs... change categories explore songs



https://pudding.cool/2024/11/love-songs/



https://pudding.cool/projects/vocabulary/



https://pudding.cool/2018/11/dearabby/

October 11: This GoFundMe group will pay your legal fees if you tattle on Trump

NBC News

October 11: Trump Debate Comment Inspires #MuslimsReportStuff, And It's Very Funny

NPR

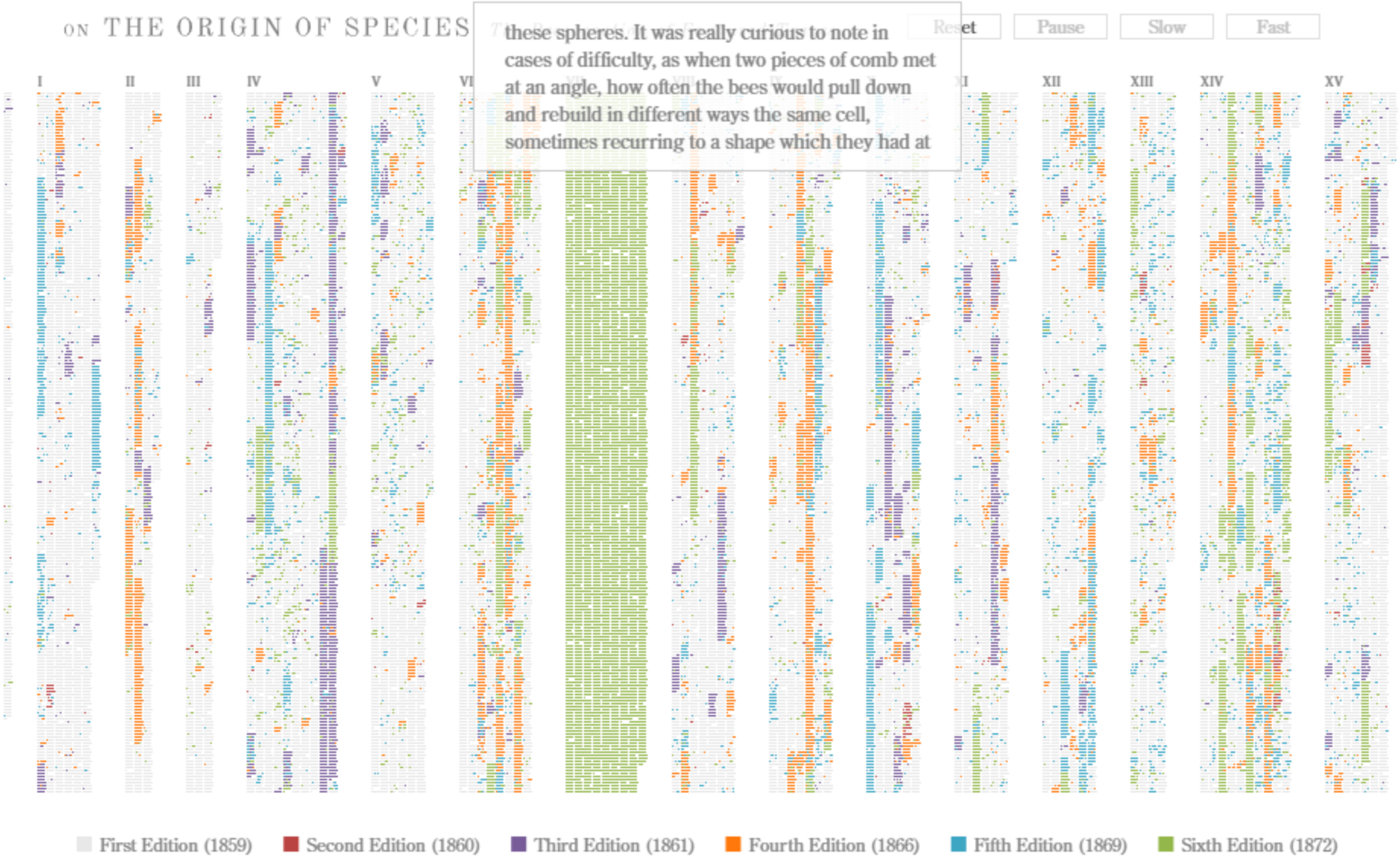
October 11: Donald Trump's Sad, Lonely Life

The New York Times

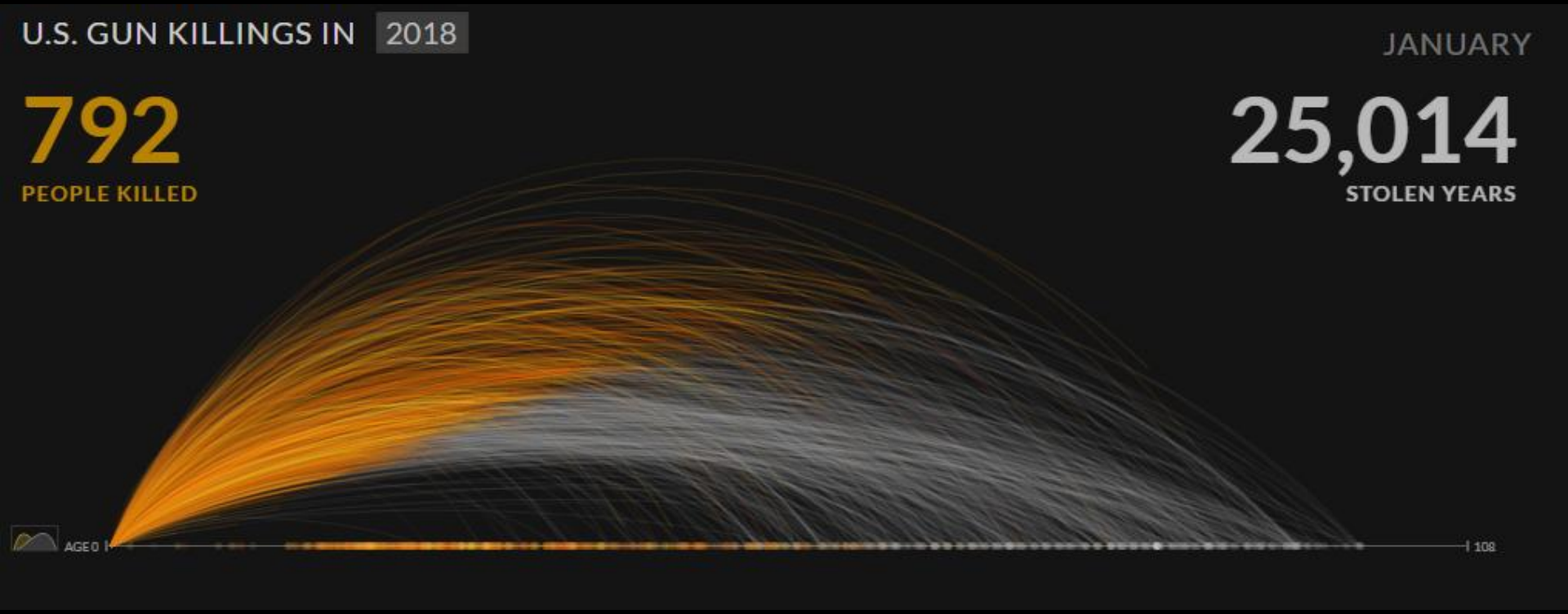
October 11: Trump declares himself free from 'shackles' — and threatens to burn the GOP to the ground

The Raw Story

https://pudding.cool/2017/03/elections/



https://benfry.com/traces/



https://guns.periscopic.com/

From domain to abstraction

- domain characterization:
  details of application domain
  - group of users, target domain, their questions & data
    - varies wildly by domain
    - must be specific enough to get traction
  - domain questions/problems
    - break down into simpler abstract tasks
- abstraction: data & task
  - map *what* and *why* into generalized terms
    - identify tasks that users wish to perform, or already do
    - find data types that will support those tasks
      - possibly transform /derive if need be

Why Analyse Tasks Abstractly?

- Transforming task descriptions from domain-specific language into abstract form allows you to reason about similarities and differences between them

Why Analyse Tasks Abstractly?

- Transforming task descriptions from domain-specific language into abstract form allows you to reason about similarities and differences between them
- Allows useful comparisons between domain situations

Why Analyse Tasks Abstractly?

- Transforming task descriptions from domain-specific language into abstract form allows you to reason about similarities and differences between them
- Allows useful comparisons between domain situations
- The apparent difference in different domains is misleading: there are a lot of similarities in the tasks people do once you strip away the surface language differences
  - e.g. an epidemiologist studying the spread of a new strain of influenza might describe her task as “contrast the prognosis of patients who were intubated in the ICU more than one month after exposure to patients hospitalized within the first week”, while a marketing manager studying sales data might use language such as “see if the southern regional sales match up with the western regional sales over the holiday season”

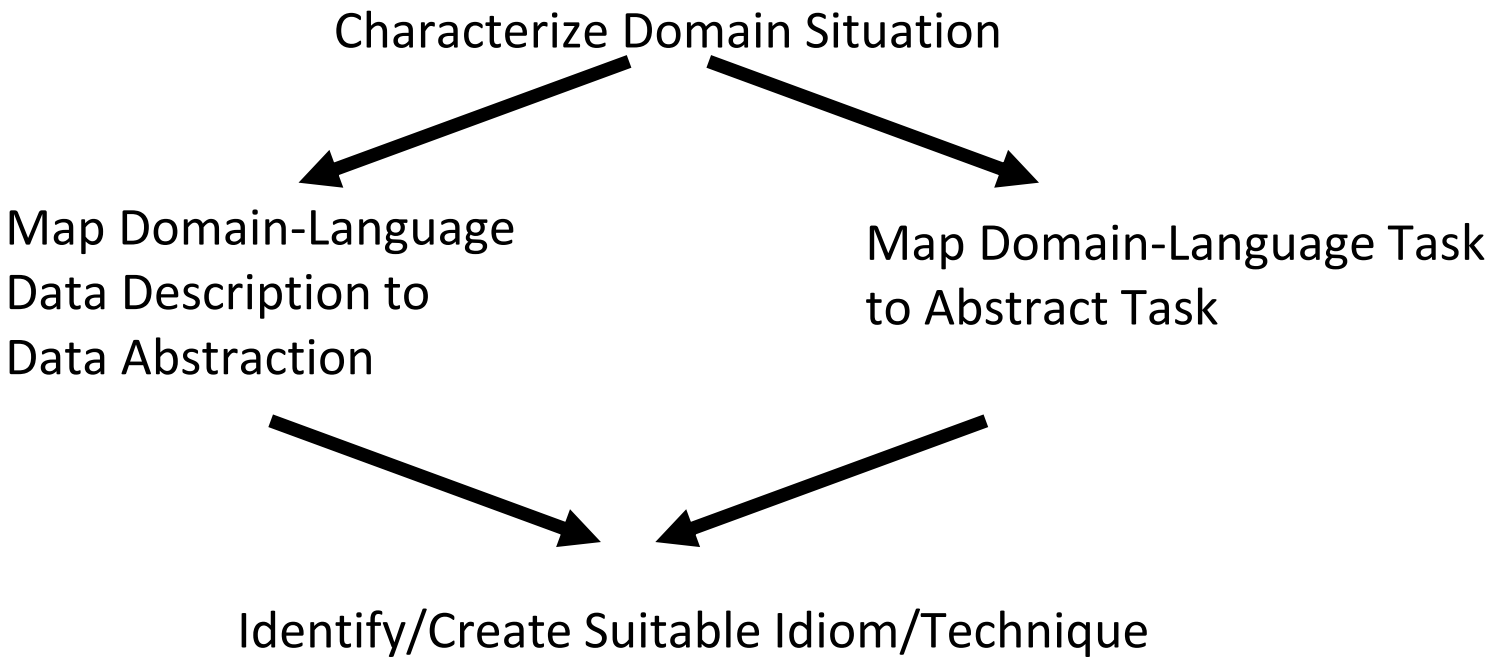
Why Analyse Tasks Abstractly?

- Transforming task descriptions from domain-specific language into abstract form allows you to reason about similarities and differences between them
- Allows useful comparisons between domain situations
- The apparent difference in different domains is misleading: there are a lot of similarities in the tasks people do once you strip away the surface language differences
  - e.g. an epidemiologist studying the spread of a new strain of influenza might describe her task as “contrast the prognosis of patients who were intubated in the ICU more than one month after exposure to patients hospitalized within the first week”, while a marketing manager studying sales data might use language such as “see if the southern regional sales match up with the western regional sales over the holiday season”
- The analysis framework has a small set of carefully chosen words to describe why people are using vis, designed to help you crisply and concisely distinguish between different goals

Why Analyse Tasks Abstractly?

- Transforming task descriptions from domain-specific language into abstract form allows you to reason about similarities and differences between them
- Allows useful comparisons between domain situations
- The apparent difference in different domains is misleading: there are a lot of similarities in the tasks people do once you strip away the surface language differences
  - e.g. an epidemiologist studying the spread of a new strain of influenza might describe her task as “contrast the prognosis of patients who were intubated in the ICU more than one month after exposure to patients hospitalized within the first week”, while a marketing manager studying sales data might use language such as “see if the southern regional sales match up with the western regional sales over the holiday season”
- The analysis framework has a small set of carefully chosen words to describe why people are using vis, designed to help you crisply and concisely distinguish between different goals
- Another important reason to analyze the task is to understand whether and how to transform the user’s original data into different forms by deriving new data

Design process



Task abstraction: Actions and targets

- very high-level pattern

- {action, target} pairs
  - *discover distribution*
  - *compare trends*
  - l*ocate outliers*
  - *browse topology*

Task abstraction: Actions and targets

- very high-level pattern

- actions
  - analyze
    - high-level choices
  - search
    - find a known/unknown item
  - query
    - find out about characteristics of item

- {action, target} pairs
  - *discover distribution*
  - *compare trends*
  - *locate outliers*
  - *browse topology*

Task abstraction: Actions and targets

- very high-level pattern

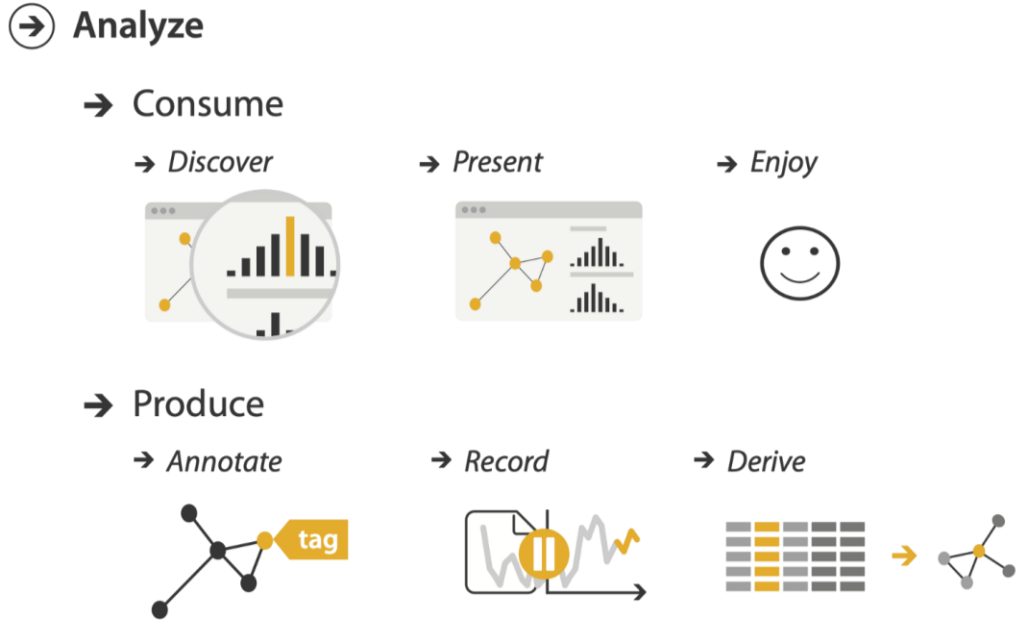
- actions
  - analyze
    - high-level choices
  - search
    - find a known/unknown item
  - query
    - find out about characteristics of item

- targets
  - what is being acted on

- {action, target} pairs
  - *discover distribution*
  - *compare trends*
  - *locate outliers*
  - *browse topology*

Actions: Analyze

- consume
  - discover vs present
    - classic split
    - aka explore vs explain
  - enjoy
    - newcomer
    - aka casual, social
- produce
  - annotate, record
  - derive
    - crucial design choice



Actions: Search

Actions: Search

- what does user know?
  - target, location

Search

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

Actions: Search

- what does user know?
  - target, location
- lookup
  - ex: word in dictionary
    - alphabetical order

Search

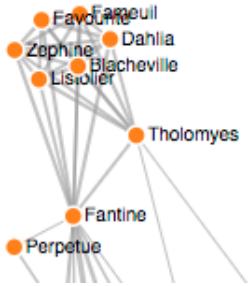
	Target known	Target unknown
Location known	 *Lookup*	 *Browse*
Location unknown	 *Locate*	 *Explore*

Actions: Search

- what does user know?
  - target, location
- lookup
  - ex: word in dictionary
    - alphabetical order
- locate
  - ex: keys in your house
  - ex: node in network

Search

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



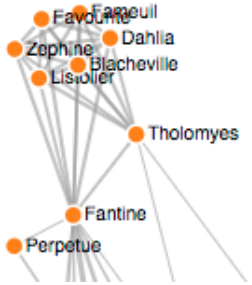
https://bl.ocks.org/heybignick/3faf257bbbbc7743bb72310d03b86ee8

Actions: Search

- what does user know?
  - target, location
- lookup
  - ex: word in dictionary
    - alphabetical order
- locate
  - ex: keys in your house
  - ex: node in network
- browse
  - ex: books in bookstore

Search

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



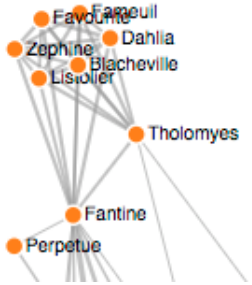
https://bl.ocks.org/heybignick/3faf257bbbbc7743bb72310d03b86ee8

Actions: Search

- what does user know?
  - target, location
- lookup
  - ex: word in dictionary
    - alphabetical order
- locate
  - ex: keys in your house
  - ex: node in network
- browse
  - ex: books in bookstore
- explore
  - ex: find cool neighborhood in new city

Search

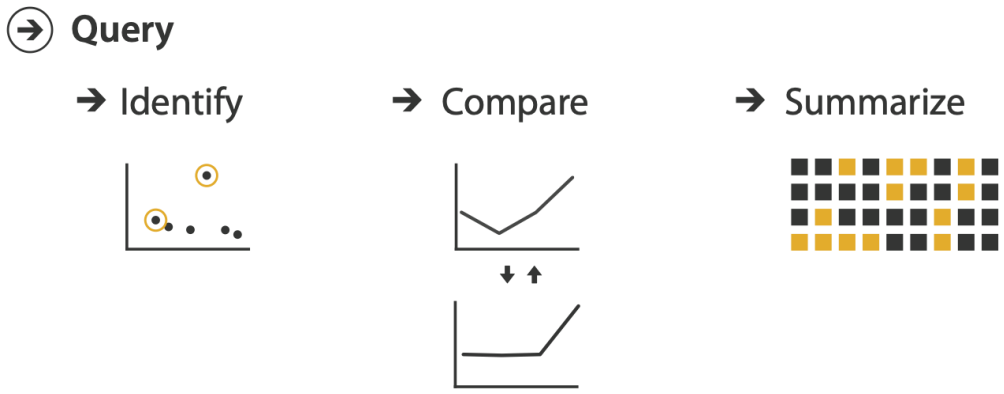
	Target known	Target unknown
Location known	*Lookup*	*Browse*
Location unknown	*Locate*	*Explore*



https://bl.ocks.org/heybignick/3faf257bbbbc7743bb72310d03b86ee8

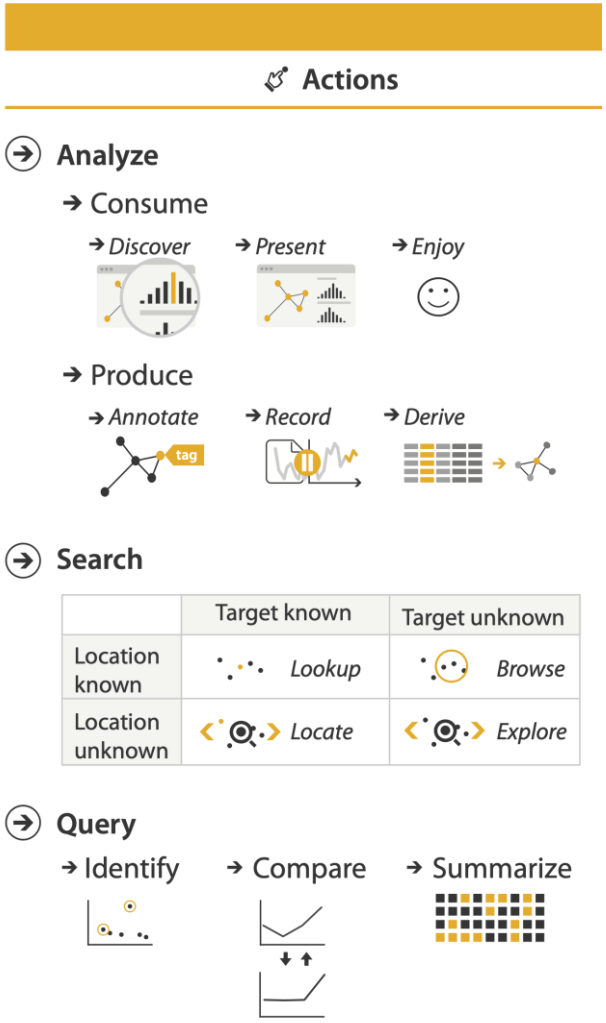
Actions: Query

- how much of the data matters?
  - one: identify
  - some: compare
  - all: summarize



Actions

- independent choices for each of these three levels
  - analyze, search, query
  - mix and match

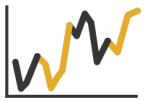
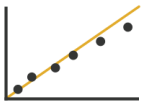
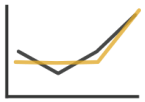


Task abstraction: Targets

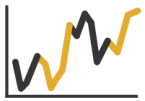
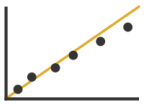
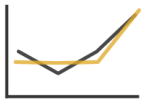
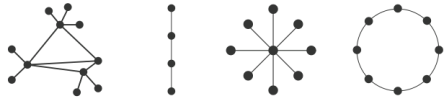
Task abstraction: Targets

- **All Data**
  - Trends
  - Outliers
  - Features

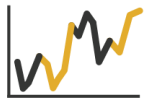
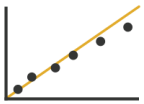
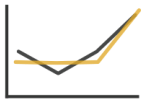
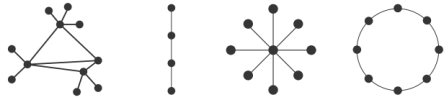
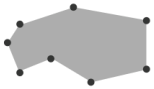
Task abstraction: Targets

- **All Data**
  - Trends 
  - Outliers 
  - Features 
- **Attributes**
  - One
    - *Distribution* 
    - *Extremes* 
  - Many
    - *Dependency* 
    - *Correlation* 
    - *Similarity* 

Task abstraction: Targets

- **All Data**
  - Trends 
  - Outliers 
  - Features 
- **Attributes**
  - One
    - *Distribution* 
    - *Extremes* 
  - Many
    - *Dependency* 
    - *Correlation* 
    - *Similarity* 
- **Network Data**
  - Topology 
  - *Paths* 

Task abstraction: Targets

- **All Data**
  - Trends 
  - Outliers 
  - Features 
- **Attributes**
  - One
    - *Distribution* 
    - *Extremes* 
  - Many
    - *Dependency* 
    - *Correlation* 
    - *Similarity* 
- **Network Data**
  - Topology 
  - *Paths* 
- **Spatial Data**
  - Shape 

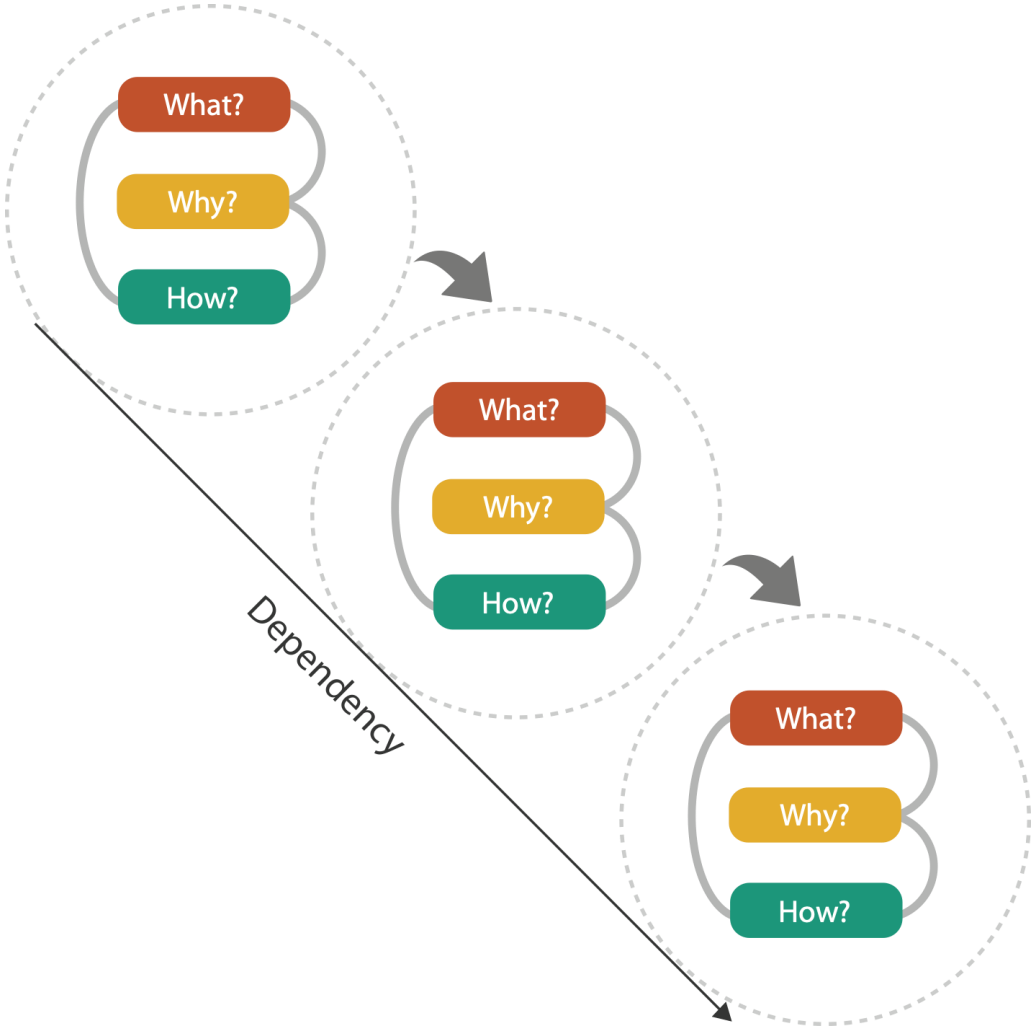
Abstraction

- these {action, target} pairs are good starting point for vocabulary
  - but sometimes you'll need more precision!

- rule of thumb
  - systematically remove all domain jargon

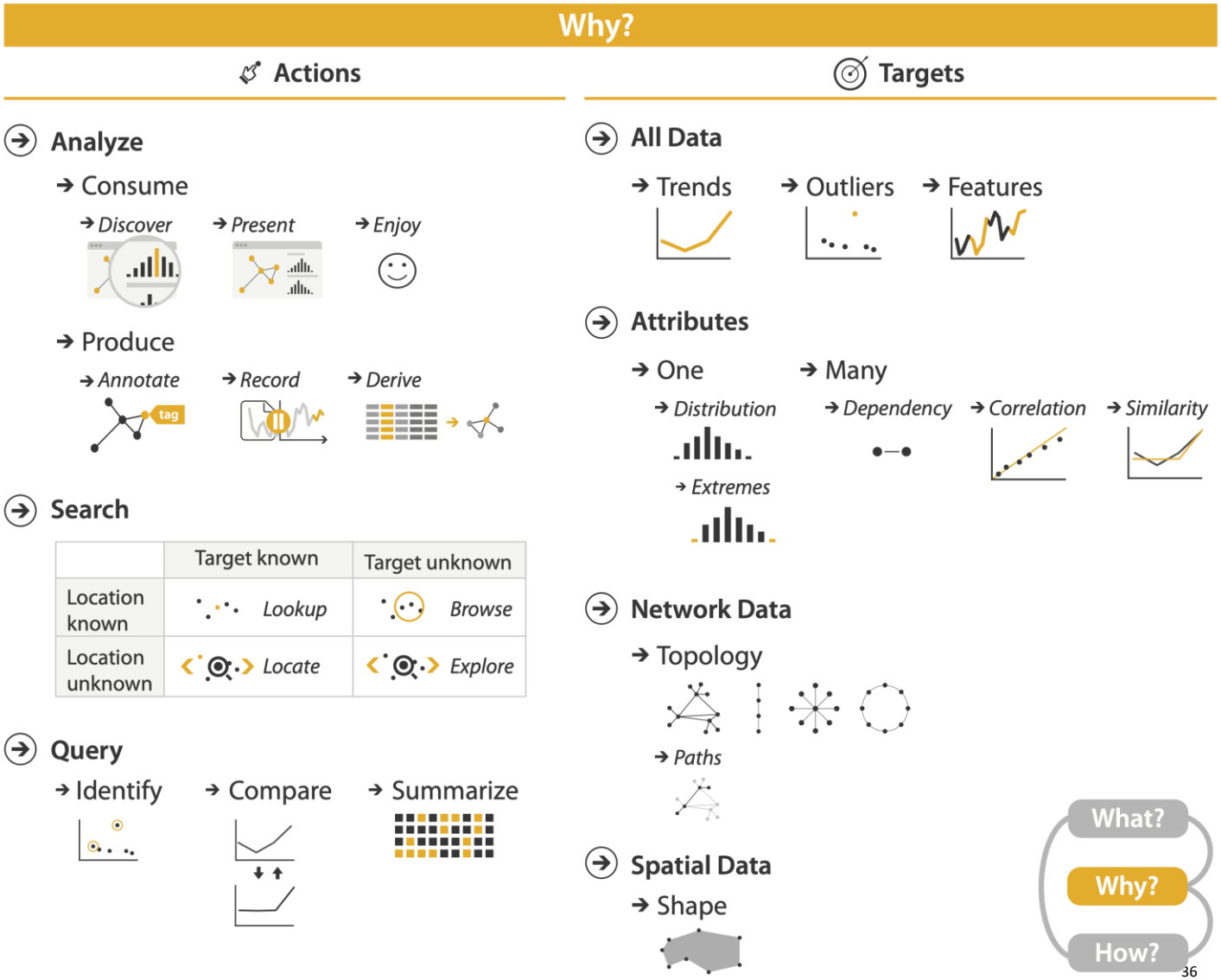
- interplay: task and data abstraction
  - need to use data abstraction within task abstraction
    - to specify your targets!
    - but task abstraction can lead you to transform the data
  - iterate back and forth
    - first pass data, first pass task, second pass data, ...

Means and ends

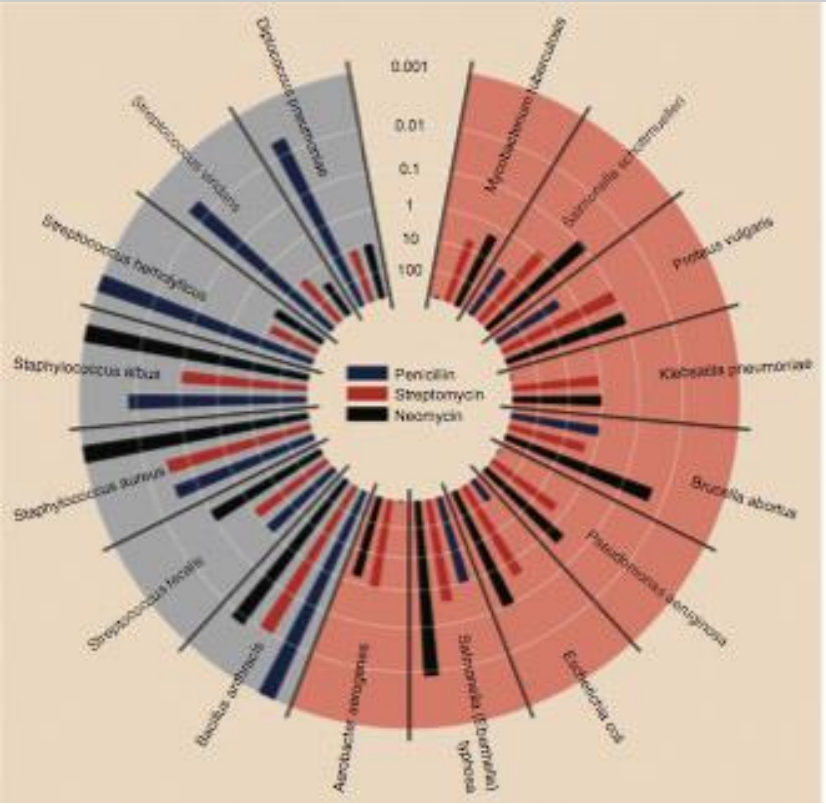


{action, target} pairs

- *discover distribution*
- *compare trends*
- *locate outliers*
- *browse topology*



Assignment 4 - Visualise Burtin's Antibiotic Dataset



Bacteria	Penicillin	Antibiotic Streptomycin	Neomycin	Gram stain
Aerobacter aerogenes	870	1	1.6	–
Brucella abortus	1	2	0.02	–
Bacillus anthracis	0.001	0.01	0.007	+
Diplococcus pneumoniae	0.005	11	10	+
Escherichia coli	100	0.4	0.1	–
Klebsiella pneumoniae	850	1.2	1	–
Mycobacterium tuberculosis	800	5	2	–
Proteus vulgaris	3	0.1	0.1	–
Pseudomonas aeruginosa	850	2	0.4	–
Salmonella (Eberthella) typhosa	1	0.4	0.008	–
Salmonella schottmuelleri	10	0.8	0.09	–
Staphylococcus albus	0.007	0.1	0.001	+
Staphylococcus aureus	0.03	0.03	0.001	+
Streptococcus fecalis	1	1	0.1	+
Streptococcus hemolyticus	0.001	14	10	+
Streptococcus viridans	0.005	10	40	+

- 3 antibiotics, penicillin, neomycin and streptomycin on 16 bacteria
- minimum concentration of the drug required to prevent the growth of the bacteria in vitro -- the minimum inhibitory concentration (MIC)
- their efficacy varied over six orders of magnitude
- scale varies from 1,000 micrograms per milliliter on the innermost ring to .001 micrograms per milliliter on the outermost
- the longer the bar, the greater the efficacy of the antibiotic.

Assignment 2 - Visualise Burtin's Antibiotic Dataset

- How do the drugs compare?
- How do the bacteria group together?

- What is produced in Nebraska?
- Where is corn produced?