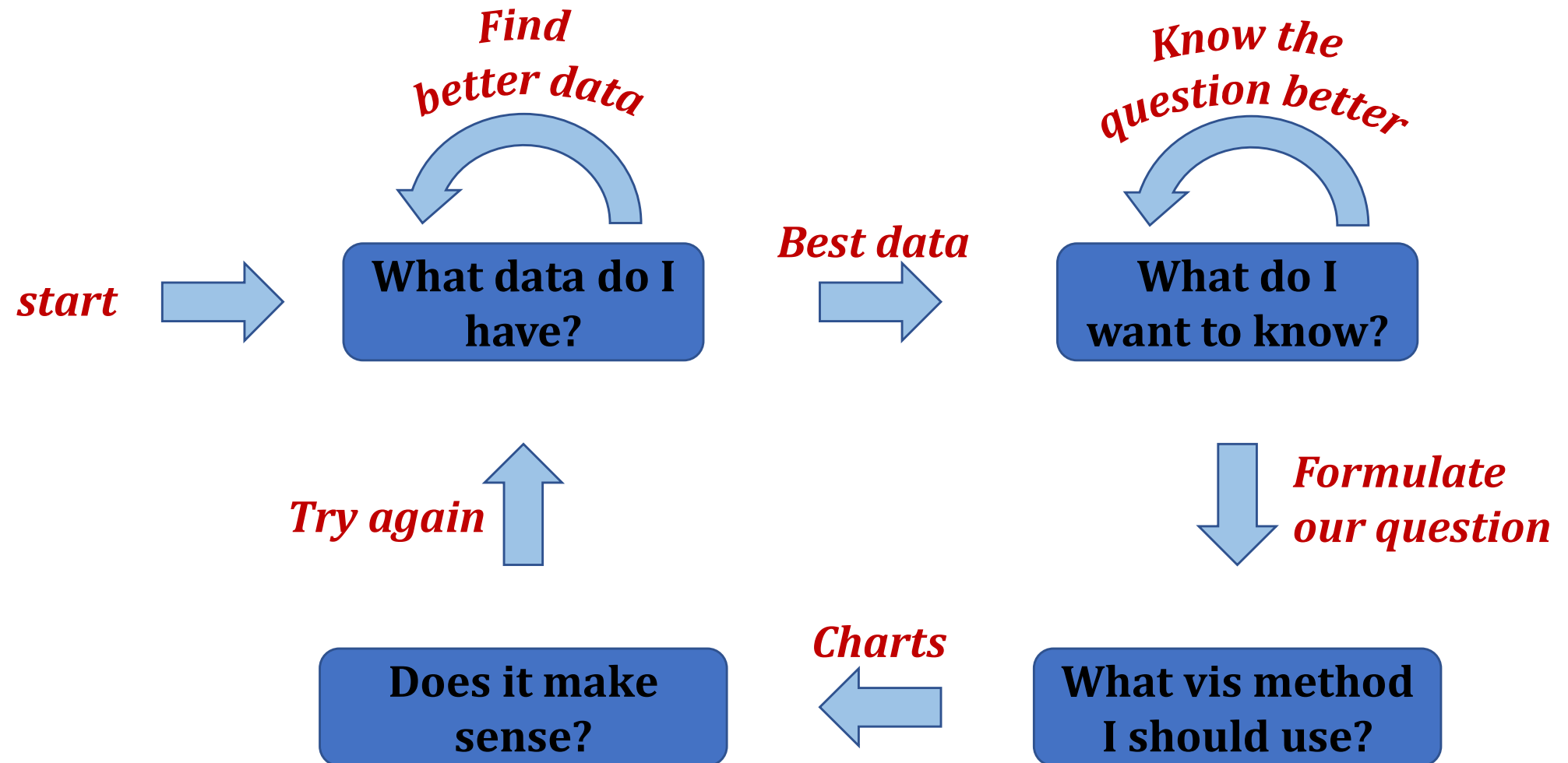


Chapter 03

Task Abstraction



Visualization workflow



Why analyze tasks abstractly?



- Consider tasks in *abstract* forms, rather than the *domain-specific* way
- Transforming task descriptions from domain-specific language into abstract form allows us to reason about *similarities* and *differences* between them
- If you don't do this kind of translation, then everything just appears to be different. That apparent difference is *misleading*

Who: **designer** or **User**

- *Designer* has built many choices
- Tool are limited, but their strength is that users are not faced with overwhelming array of design choices
- The breadth of choices is both a strength and a limitation:
 - *Users* have a lot of power, but they also may make ineffective choices if they do not have a deep understanding of many vis design issues

Actions

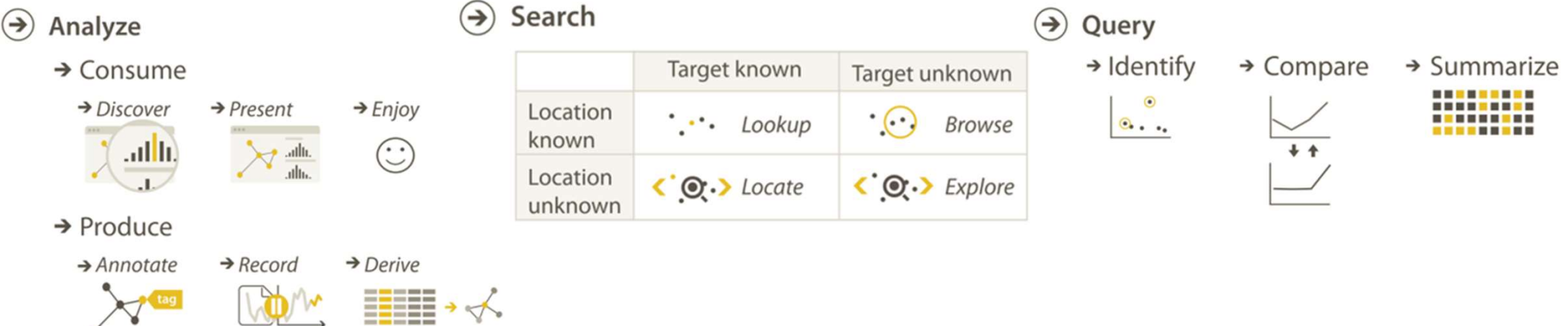
- Three levels of actions that define *user goals*

action	High-level: Analyze	Consume	Discover, present, enjoy
		Produce	Annotate, record, derive
	Mid-level: Search	Lookup, location, browse, explore	
	Low-level: Query	Identify, compare, summarize	


Actions

Actions

Actions



Actions - Analyze



- We want to *analyze* the data using a visualization tool
- Two possible goals
 - *Consume* existing information (common)
 - *Discover* something new
 - *Present* information that is already completely understood
 - For users to *enjoy* a vis to indulge their casual interests in a topic

Actions - Analyze



- Two possible goals
 - Actively *produce* new information. There are three kinds of produce goals:
 - *Annotate* – refers to the addition of graphical or textual annotations
 - *Record* – refers to save or capture visualization elements as persistent artifacts
 - *Derive* – refers to producing new data elements based on the existing ones

Actions - Search

- All the *analyze* cases require the user to *search* for elements of interests within the visualization as a mid-level goal.
- There are four search alternatives

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





Search - Lookup

- The search type is *lookup* if we know both
 - What we are looking for (target)
 - Where it is (location)

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

Search - Locate

- The search type is *Locate* if we know:
 - What we are looking for (target)
 - But don't know where it is
- This means that we must find out where it is

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

Search - Browse

- The search type is *Browse*
 - If we do not know what we are looking for (only know its characteristics)
 - But do know where it is
- This means that we must find out where it is

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

Search - Explore

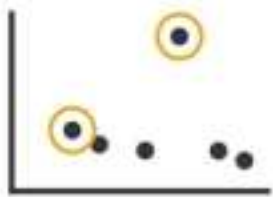
- The search type is *Explore* if we do not know
 - What we are looking for
 - Where it is
- We may search for outliers in a scatterplot, or anomalous spikes in a time-series

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

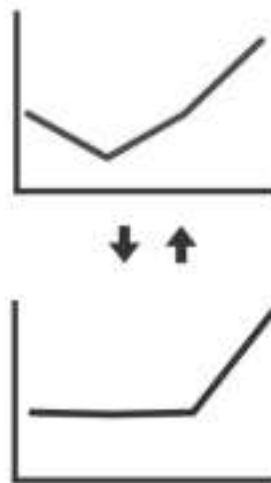
Actions - Query

- Once a set of targets for a search have been identified, a low-level user goal is to query these target at one of three scopes: *identify*, *compare* and *summarize*

→ Identify



→ Compare

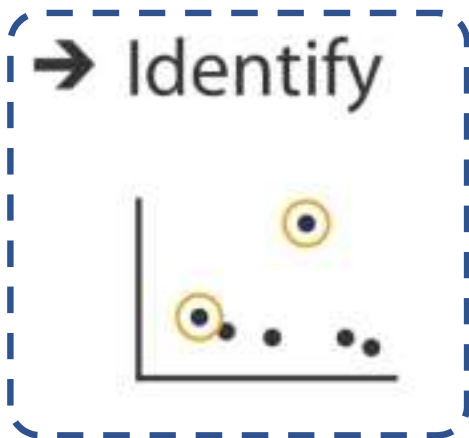


→ Summarize

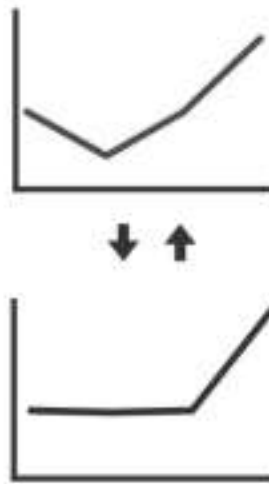


Actions - Query

- The scope of *identify* is a *single target*.
- If a *search* returns known targets, then *identify* returns their characteristics.



→ Compare



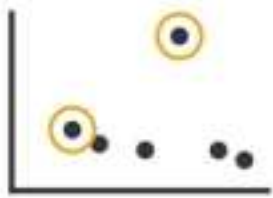
→ Summarize



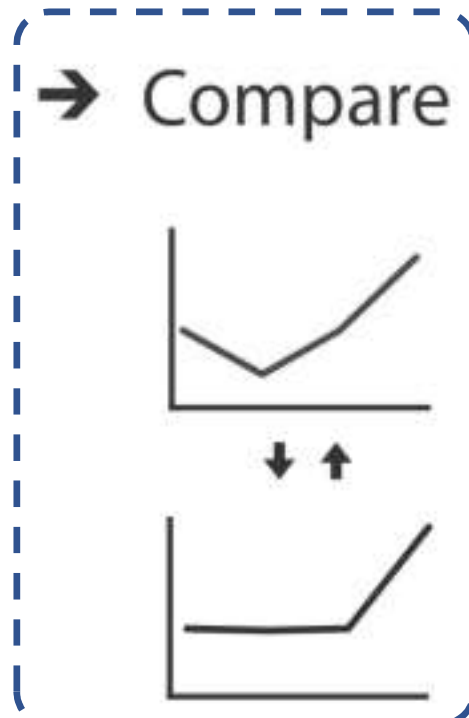
Actions - Query

- The scope of *compare* is *multiple targets*.
- *Comparison tasks* are typically more difficult than *identify tasks*, and usually require more sophisticated mechanisms to support the user

→ Identify



→ Compare



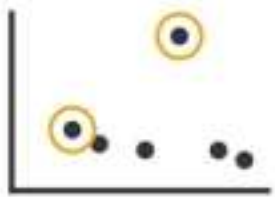
→ Summarize



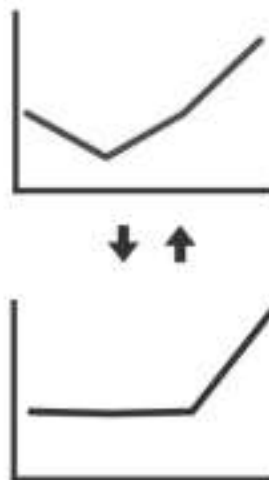
Actions - Query

- The scope of *summarize*, also known as *overview*, is *all possible targets*.
- It means providing a comprehensive view of everything, which is extremely common in visualization

→ Identify



→ Compare



→ Summarize

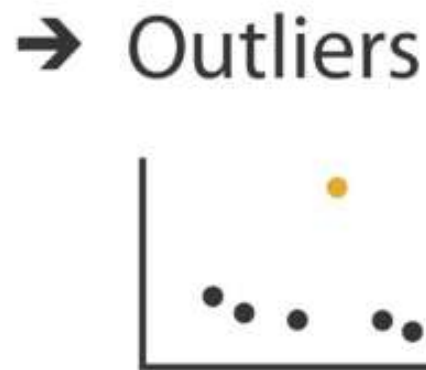
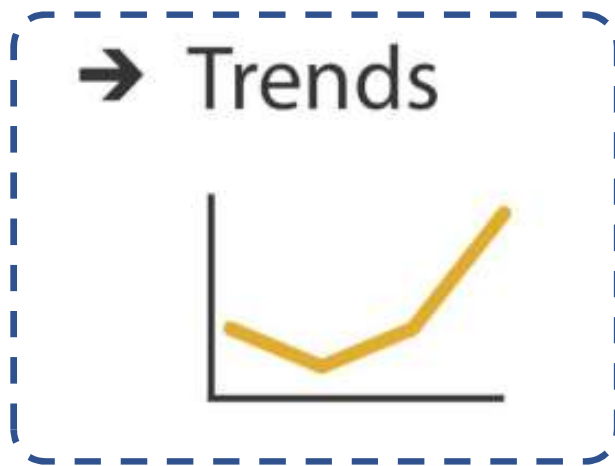


Target – **for all data**

- The actions discussed earlier refer to a *target*, meaning some aspect of the data that is of interest to the user.
- The idea of a *target* is very explicit with *search* and *query* actions.
- Three high-level targets are relevant for all kind of data: *trends*, *outliers*, and *features*

Target – **for all data**

- A *trend* is a high-level characterization of a pattern in the data.
- A trend may be *increases, decreases, peaks, skewness, plateau*, or any pattern.



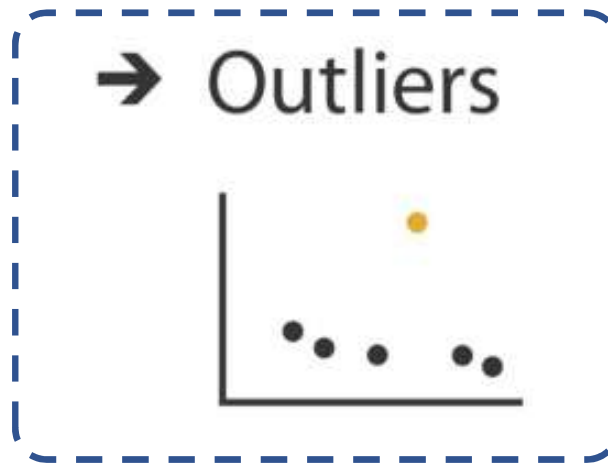
Target – **for all data**

- Some data will not fit well in a global picture of the trend or the pattern.
- These data elements are the *outliers*.
- More accurately, an *outlier* is an observation point that is distant from other observations.

→ Trends



→ Outliers



→ Features



Target – **for all data**

- It is difficult to define the meaning of *features*.
- Intuitively, a feature is task dependent, meaning any particular structures of interest..

→ Trends



→ Outliers



→ Features



Target – for attributes



- *Attributes* are specific properties that are visually encoded.
- The lowest-level target for an attribute is to find an individual value.
- Common target of interest is to find the extremes (i.e., minimum and maximum).
- A high-level scope target is the distribution of all values for an attribute

Target – for attributes

- Some targets may have the scope of multiple attributes: *dependencies*, *correlations* and *similarities* between attributes.

→ One

→ Distribution



→ Extremes



→ Many

→ Dependency



→ Correlation



→ Similarity



Targets – for Specific Types

- Some targets relate to specific types of datasets.
- The fundamental target with *network* data is to understand the structure of these interconnections (i.e., *topology*). A more specific topological target is a *path* of one or more links that connects two nodes.
- Whether a network is planar may also be important.
- For *spatial* data, understanding/comparing the geometric *shape* is a common target

User visualization

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



What?

Why?

How?

How to Design Vis. Idioms?



- There are in general four choices:
 - Encode
 - Manipulate
 - Facet
 - Reduce

How to Design Vis. Idioms?

- *Encode*: The way of encoding data within a view has five choices for how to arrange data spatially:

- Express values
- Separate, order, and align regions
- use given spatial data.

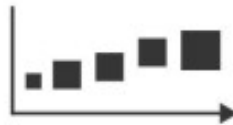
→ Express



→ Separate



→ Order



→ Align



→ Use



How to Design Vis. Idioms?

- *Encode*: Also includes how to map data (from *categorical* and *ordered* attributes) with all the non-spatial visual channels (e.g., color, size, angle, shape, etc.)

→ Color

→ Hue



→ Saturation



→ Luminance



→ Shape



→ Size, Angle, Curvature, ...



→ Motion

Direction, Rate, Frequency, ...



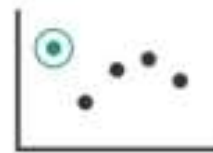
How to Design Vis. Idioms?

- *Manipulate*: This family has the choices of changing any aspect of the view, selecting elements from within a view, and navigating to change the viewpoint within the view.

➔ Change



➔ Select



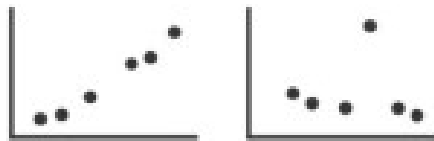
➔ Navigate



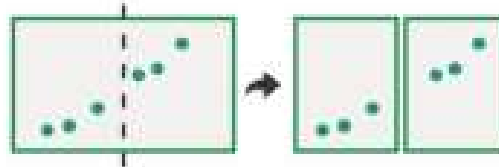
How to Design Vis. Idioms?

- *Facet*: The family of how to facet data between views has choices for how to juxtapose and coordinate multiple views, how to partition data between views, and how to superimpose layers on top of each other.

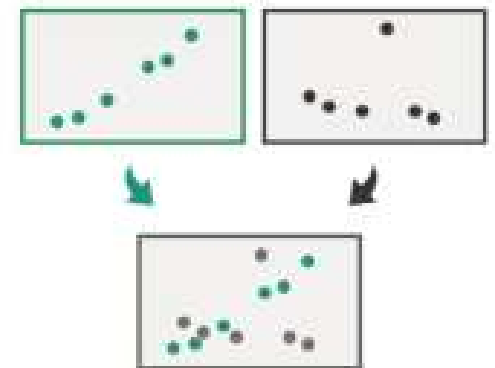
➔ Juxtapose



➔ Partition



➔ Superimpose



How to Design Vis. Idioms?

- *Reduce*: The family of how to reduce the data shown has the options of filter data away, aggregate many data elements together, and embed focus and context information together within a single view.

➔ Filter



➔ Aggregate



➔ Embed



Hết!